UNIVERSITY OF OSLO
Department of informatics

An educational system development game

Master thesis

60 credits

Eirik Molnes
Jørgen Stikbakke

2. May 2008

Abstract

Educational computer games have been around for a long time. However, the work that is being done on educational computer games is mainly focused on an elementary school level. We feel that it is high time that educational computer games took the step into universities.

To test the potential of using such games at a university level, we decided to develop our own game. In order to find out more about the important aspects of educational game design, we explored the research done within the field. Based on this research we decided to base our game on the old classic Jones in the Fast Lane. We chose this game type because it is possible to balance the important elements of motivation, entertainment and learning. The game featured educational content from one of the beginner courses for system development at the University of Oslo, namely Inf1050. This course covers a lot of the basic aspects of system development like system development strategies, relational databases and data models.

We tested the game on students attending the INF1050 course in the spring of 2008. Our test results showed that those students who responded are excited about the use of educational computer games. They expressed that they potentially could learn a lot if such games were used as a supplement to the regular curriculum. We also see that there is a huge potential for using educational computer games to increase the time students spend on their studies.

Through the development of this game and our testing, we see that the most important factors in an educational computer game are entertainment and motivation. The entertainment needs to be present in order to keep the attention of the player, while motivation encourage the player to play the game.

We conclude that if one merge the aspects of motivation, entertainment and education, it is possible to make educational games that students at a university level would enjoy and learn knowledge from while playing.

Acknowledgements

First we would like to thank our supervisor Jens Kaasbøll. We appreciate all the time and effort he put into helping us with this thesis. We would also like to thank Odd Aurmo for his contribution with correction and text analysis and Gerhard Skagestein for the original idea to this thesis.

Thanks to Morten for all the laughs, Øystein for all the critisism and Irene for her patience.

We dedicate this thesis to our moms who carried us for 9 months. Without their late night milk and home baked cookies we never would have finished.

Table of Contents

Illustration Index

Index of Tables

Chapter 1
Introduction

1.1 Thesis

Educational computer games have been around for a long time. The early versions consisted of simple games meant to teach language or typing skills. Over the years educational games have evolved and now you can find educational computer games for almost any topic. These are mainly made by idealistic organizations (Nobelprize.org - Educational Outreach Program, 2008), and see surprisingly little use in schools. Mark Prensky of games2train.com (",2008) concluded in an interview with technewsworld.com (Korzeniowski, 2007) that: "The students are ready for educational video games but the educators are not. That outlook will change slowly, and games will play a significant role in the classroom." The work that is being done on educational computer games however is mainly focused on an elementary school level. There seems to still be a dogma surrounding computer games, in that they are considered amongst many to only be suitable for children. This despite the fact that the Entertainment Software Association's (", 2008a) research paper «2007 Essential Facts About the Computer and Video Game Industry» (Entertainment Software Association, 2008b) shows that the average age for computer gamers was 33 in 2007, a number that keeps going up year by year.

In light of the fact that computer gamers are getting older, and that educational computer games are still largely made for children, we feel that it is high time that educational computer games took the step into universities.

We intend to make an educational computer game to test the motivational benefits of such games when used at a university level. In order for our educational game to be successful we will explore the research done within the field to gain a better knowledge of the important aspects of educational game design. Based on this research we will decide upon a game type and how to implement the educational content into the game.

We will fill the game with material from one of the beginner courses for system development at our university, namely INF1050. This course covers a lot of the basic aspects of system development. We will explore the main topics in INF1050 and identify the key topics that are best suited to add into our game.

To find out what the motivational benefits of such games are, we will test our game on students attending the INF1050 course. Our work will focus on the motivational aspects of educational computer games and how the students perceive the potential gain from playing such games. We will try to show that educational computer games can motivate students to spend more time on their studies. Based on the assumption that more time spent equals more knowledge learned, this should indicate the value of using educational computer games.

Research questions:
1. Is it possible to make an educational computer game for a specific course at a university which increases the students' motivation towards learning and the time they spend on studies? Will this game contribute to improve their learning?

2. Can the knowledge we learn from our project say anything about educational computer games in general? If so, what does educational game designers need to focus on in the future, and how can this be used to make educational game for students at a university level?

Our hope is that by providing empirical evidence that support the use of educational computer games at a university level, we can contribute to shift the focus of developers to not only include games for youth, but students of all ages.

1.2 Contribution

There have been limited testing of educational computer games on a university level. We hope that by creating an educational computer game for a specific course at a university level, we can accommodate sufficient data to motivate developers to become involved in creating educational computer games for universities. We believe that the data should prove sufficient to be applied on a general basis, even though the data we collect will be from a single game, with content based on a single course.

Our focus will be on how the motivational aspects of games can help motivate students to use more time on their studies, and thus learn more. Studies done on Norwegian students by

Statistisk Sentral Byrå (Løwe and Sæther, 2007) showed that Norwegian students on a university level used an average of 30 hours per week on their studies. The former Norwegian Minister of Education Øystein Djupedahl (Stenersen and Pedersen, 2007) and The Norwegian Association of Students (The Norwegian Association of Students, 2007) both gave statements regarding the study and commented that the time students spend on their studies was insufficient. Adding additional motivating factors like educational computer games to higher learning might be one of the solutions to getting full time students. That is still one of the main goals of the Norwegian Ministry of Education and Research (Djupedal, 2007).

1.3 Outline of the thesis

In chapter two we give an overview of some of the earlier work that is done on educational computer games. We briefly present key papers and determine their results. To conclude the chapter we will describe how this research relates to our thesis, and how our contribution can supplement it.

Chapter three focuses on what an educational computer game is, why they can help motivate students to learn more. We list the different types of educational computer games we looked at when choosing our implementation and explain the benefits and drawbacksto each variation. To conclude the chapter we will say what type of implementation we decided on and why this was the natural choice for us.

Chapter four defines what System Development is in regards to the curriculum of INF1050. We identify the key points of the course and suggest ways to test this knowledge with different assignment implementations. Based on the implementational difficulty of these assignments we chose what parts of the INF1050 curriculum to implement. Further we present the design characteristics and functional requirements for our game type. We base our choices on what we learned from the earlier work presented in chapter two, and how to best merge the educational content with the entertainment value of the game.

Chapter five takes a look at the technologies we used to develop our game, and manage our cooperative work. We explain why our choice of programming language fell on Java. To conclude the chapter we look into existing technology and why we chose to develop a game from scratch.

Chapter six gives a detailed description of the design of our game. In this chapter we identify all the different elements that make up our game, and how these elements cover the design aspects needed to make the game entertaining. In addition we give a short walkthrough of the gameplay.

In chapter seven we explain our choice of implementation. We present an overview of the class diagram for the game, and break this diagram into its parts to explain how all the elements of the game work in relation to each other. We also look into the file system related to our system, and why we chose the different forms of data storage that we did.

In chapter eight we present the two sets of tests we did on users. The first test round was on a small user group to test our implementation and make corrections according to the feedback. The second test round was the main test where we tested our finished game on the students of INF1050. We outline our goals for both tests, explain the methods we used and how the tests actually were conducted. To conclude the chapter we present our finding for both these tests, and what conclusions we drew from them.

Chapter nine discusses whether we managed to achieve what we set out to do in this thesis. First we look at how our game met the requirements we set forth. Specifically if the game met the design and functional requirements, and whether or not it was entertaining enough to motivate the testers. We also look into what could have been done differently during our thesis work. Finally we discuss whether or not our overall goal was reached.

In chapter ten we give our conclusion of this thesis. We give a short summary of all our work, and what our results were. We go on to discuss possible future work that can be done to our game and the research on educational computer games on a university level. We look into alternate use for our software, additional testing that may be done, and how the software can be expanded upon.

Chapter 2
Background

2.1 Why computer games as an educational tool

In this chapter we will look at earlier work done within the field of educational computer games and educational games in general. The main focus will be on the motivational aspects of games, and how they can improve the learning process. Through these papers we will identify the key aspects of games that make them such great motivational tools. Later in the chapter we will compare the different papers and give our thoughts on what we learned. To conclude the chapter we look into how this knowledge can be used in our thesis.

2.1.1 Learning by design: good video games as learning machines

In his paper "*Learning by design: good video games as learning machines*" (Gee, 2005) James Paul Gee focuses on the learning aspects of successful computer games. He states that the learning process these games encourage is the primary reason they hold such an interest for gamers.

Yet, when people think about computer games, they think about entertainment, and when people think about learning, they think about work. This is seen as a key point. As Gee states it, people are used to learning environments where learning is forced upon them. Computer games however break this trend, as they involve a great deal of learning, yet it is seen as a leisure activity. So the question that Gee asks is: How do game designers manage to get players to learn their long and complex games, and even pay to do so?

Gee claims that learning is in itself, under the right conditions, biologically motivating for human beings, much like sex. People need stimulation, and feel a great sense of accomplishment when they complete new and challenging tasks. Games have, through

Darwinian methods, found a way to create such optimal conditions for learning. Gee presents examples of successful computer games, and how each one of them presents a balanced learning curve for the gamer. The games that fail to challenge in this way, are also the games that land at mediocrity.

Gee concludes that because the gaming industry is so competitive, and the learning aspects of games is so closely tied to their success, good game designers have become experts on the theories of learning. Further Gee tries to pinpoint and categorize the learning elements of computer games, and why they compel gamers. This is the main part of his paper, as he feels identifying the key points that make up a compelling game, gives knowledge that can be used to further game development and improve general education. We will present a shortened list of his key points below.

Empowered learner:

- Co-design - Players feel their actions and decisions directly affect the world the game world.
- Customize - Players have the ability to customize the difficulty to their level.
- Identity - Players identify themselves with the characters in the game.
- Manipulation - Players get a sense of accomplishment from advanced character manipulation.

Problem solving:

- Well-Order Problems - A natural difficulty progression. Each successive problem gives you basic skills that are used in later more difficult problems.
- Pleasantly Frustrating - Hard, but doable. Players get evidence that they are progressing when faced with a particularly challenging problem.
- Cycles of Expertise - Skills are practised, mastered and then tested, before a new skill-set is introduced.
- Information “On Demand” and “Just in Time” - This is an important principle when giving information in a learning environment. People are generally bad at storing verbal information, and it should thus only be given when it is needed.
- Fish tanks - A simplified game environment for learning purposes
- Sandboxes - A no-risk environment, introduced in a game to stem the learning curve.
- Skills and Strategies - Giving players a purpose to train skills, so it doesn’t feel meaningless.

Understanding:

- System Thinking - People learn best when their decisions and actions fit into a broader system, giving it purpose.
- Meaning as action image - Words and concepts should be clearly tied to actions in the world.

Gee concludes by commenting how many of these principles are found in good computer games, and how few of them you find in educational games and normal learning environments. He suggests that educators should take a deeper look at the lessons learned in computer game design.

2.1.2 Motivating Learners in Educational Computer Games

In "*Motivating Learners in Educational Computer Games*" (Tuzun, 2003) Hakan Tuzun identifies the motivational elements of an online multiplayer educational computer game. The game he chose for his study is called Quest Atlantis (QA), which he himself has co-developed. QA is a virtual learning environment designed for children. In the game children cooperate in the online world of Atlantis to try to save the island from destruction. Tuzun studied one school's use of QA and sought to identify the factors that made the game motivating and fun for the children. From the information he got through interviews, observation and the study of QA database information, Tuzun arrived at thirteen categories for motivation.

- Identity presentation: The kids identify with the game through their avatar's choices, homepages and user names. This empowers them within the game.
- Social interaction: Satisfaction from helping and competing with the other users.
- Playing: Controlling the character in the game-world and exploring QA was deemed fun.
- Learning: The kids enjoyed the educational value of QA. They emphasized that learning in QA was fun.
- Ownership and control: QA is based on 'Participatory Design', as such the kids felt that the game was partly 'theirs'.
- Fantasy: The fantasy element of QA, the immersion of the game-world, was a

motivational factor to the kids.

- Immersive context: The QA game came with real items such as QA trading cards and posters, which added to the impressiveness of the game.
- Curiosity: The unknown elements of the game made the kids want to play more often, to discover new secrets within the game.
- Creativity: QA let the kids build virtual houses within the game-world. A lot of the kids mentioned how fun this form of creativity felt.
- Achievement: Completing quests in QA gave them a sense of accomplishment
- Reward: They liked the in-game rewards, such as trading cards.
- Uniqueness: They enjoyed the uniqueness of the game, as they had not had any similar gaming experiences.
- Context of support: Different play experience at different locations.

Tuzun concludes that a lot of the kids had a difficult time separating playing and learning when they were occupied with QA. This provided the kids with a positive attitude towards learning.

2.1.3 The Curse of Monkey Island: Holding the Attention of Students Weaned on Computers

In «*The Curse of Monkey Island: Holding the Attention of Students Weaned on Computers*» (Ladd, 2006) Brian C. Ladd describes how they teach computer science in an introductory course at St. Lawrence University. This is done by having the students make text based games (or interactive fiction), as opposed to the normal "Hello World" scripts. The basic concept was that since youth today are so involved with computer games, designing such games would be more motivational to them.

After a short period of basic language assignments, and a brief introduction to interactive fiction, the students are presented with the main assignment of the course. The assignment is to design an interactive fiction, and has few specifics. This allows the students to use their own imagination to choose the content of the game. This is one of the huge motivational components of these assignments, as students can, within loose boundaries, design any game they want.

One of the first observations Ladd made, was that student interaction shot through the roof as they introduced this assignment. As opposed to the normal "Hello World" scripts they had

been using, this assignment created excitement among the students. They discussed the various ideas for unique implementations and "special features". Another positive by-product they got from introducing this assignment was that plagiarism went down, as student creativity went up. Since the students felt compelled to create a unique game, they had little to gain from copying other people's code.

The students go through several phases of game design as the teachers guide them along, ensuring that they do not over complicate the assignment. Through the process the students also get some basic knowledge about computer game design, as creating a game that is fun to play is a major part of the motivational factor of the assignment.

Ladd reports great success with the projects after having used it for three years. Course evaluation suggests that the students are more satisfied with the course now, than before the introduction of the project. The students rarely drop out from the course anymore, one in the three year period. Student engagement seems to have improved, and the average student uses more time on the course than they used to. Ladd reports success across the board, and advocates strongly for other faculties to follow their example.

2.1.4 Heuristics for Designing Instructional Computer Games

In "*What makes things fun to learn? Heuristics for designing instructional computer games*" (Malone, 1980) Malone presents his intuitions on what makes a computer game fun, based on the experiments and theory of his earlier work. His primary goal with this paper is to provide some basic guidelines for good game design of instructional computer games. Even though he focuses on instructional computer games, he does not focus on what makes them instructional. He rather focuses on what makes them entertaining.

He organizes the basics of good computer game design into three categories: challenge, fantasy and curiosity. He then goes on to describe each in detail.

Challenge:

The most basic element of challenge is to provide a goal whose attainment is uncertain. In his studies he found that the single most sought feature in a computer game was a goal. A good and well defined goal gives a gamer a "purpose". Using a skill should be a means of achieving a goal, but should not be the goal itself. Malone presents several practices for

implementing goals in computer games, such as keeping them obvious, and anchoring them in practical real world problems or a specific fantasy. Users also need to be able to tell whether they are getting closer to a goal.

Malone states that another aspect of providing a challenge, is keeping the outcome uncertain. A game quickly gets boring if it is certain whether you will win or loose. Malone proposes four ways of keeping the outcome uncertain, which all focus on keeping the level of difficulty as close to the players ability as possible. The first is providing a variable level of difficulty, either determined automatically by the game, an opponent or selected by the user. Another way of keeping the game challenging, but still doable, is by providing multiple level goals where advanced players can seek to perfect their gameplay, while average players still have the opportunity to progress within the game. The last two ways of keeping the outcome uncertain, is keeping the player in the dark about certain key gameplay mechanics and introducing randomness.

The next major point Malone presents as a driving force in gaming is the self-esteem reassurance well designed games give. Like accomplishing any other challenging task, completing a challenging computer game gives the player a boost to their self-esteem. Malone points out that it is important to find the thin line between rewarding feedback of progress and providing a sufficient challenge to the player.

Fantasy:

Malone goes on to debate the importance of a fantasy setting in computer games. In a game with a fantasy setting, the objects and game world is represented by images that support some form of fantasy. They try to resemble the real or fictional setting, unlike non-fantasy games which contain only abstract symbols.

Malone divides the fantasy aspect into extrinsic and intrinsic fantasies. In extrinsic fantasy based games, the fantasy is based on the player performance. That is to say, the input the player provides to the game determines the outcome of the fantasy. In intrinsic fantasies, the player performance is also based on the fantasy. Thus the player's knowledge about the fantasy setting will affect his or her performance.

The main benefit of providing a fantasy setting, is that is creates an arbitrary motivational

factor to support the game goal. Malone points out two ways of accomplishing this. Either by having the player accomplish some task, like building a skyscraper, or by avoiding some catastrophe. Both of these fantasy settings provide a reason for the player to use his skill in the game, but in Malone's opinion, intrinsic fantasies are both more interesting and instructional than extrinsic fantasies. Malone goes on to suggest that fantasy based games might satisfy some emotional needs in the players, and that that is the main reason for their success.

Curiosity:
The next motivational factor presented is curiosity. Malone states that computer games can invoke the player's curiosity by providing an "optimal level of informational complexity". When this is done correctly, the game seems novel and surprising, but not completely incomprehensible to the player. Malone divides curiosity related to computer games into sensory curiosity and cognitive curiosity. Sensory curiosity is when the player's curiosity is sparked through the use of sound or imagery. Sensory curiosity can be used to support the fantasy setting, or a reward feature of the game itself. When used as a reward feature, the players are presented with tidbit of information when completing certain tasks within the game. Cognitive curiosity on the other hand is based on the theory of the individuals desire to bring their knowledge into structure of completeness, consistency and parsimony. By providing the players with information that is initially incomplete, inconsistent and unparsimonious, the player is motivated to progress in the game to complete the knowledge structure.

After providing the list of motivational aspects of computer games, Malone suggest that it can be used as a tool when developing or improving computer games. He states that, in general, the more of these features that are incorporated in a computer game, the more motivating it will be to the users. He goes on and presents examples where existing games could be improved by using his list and suggestions on how to do so. Lastly he claims, as an oddity, that these principles also apply to computer programming and to some degree real life.

2.2 Designing with motivation in mind

As we design a computer game for educational purposes, it is important not to forget the motivational aspects of such a game. As we present in this section, the focus must be on the

motivational, rather than the educational, aspects of the game. If no one is willing to play the game, then nothing is learned from it.

Educational games are not a new venture. They have coexisted with traditional computer games for a while now. Educational game design however has not been a particularly fortuitous venture. It is a field that has been prone to failure. Even though research on the reasons for the failure of some educational games drown in the vast sea of self glorifying propaganda, is obvious, from the lack of successful educational computer games alone, that there is room for improvement in the field. As suggested by James Paul Gee and Thomas W Malone, designers of educational computer games can learn a lot by looking at what makes competitive games successful. The most successful computer games are, as stated by Gee, learning machines in their own right, so why should it necessarily be so hard to design games for educational purposes?

One of the first things that come to mind, from our own experience of earlier educational games, is that they forget the motivational aspects that have to be there to keep the player focused and entertained. When the focus is lost, little is learned. The emphasis in educational computer game design must thus put entertainment first and let the educational aspects come as a natural second.

As we explore educational game design, it is important for us to discover the basic concepts that captivate and motivate players of computer games. As can be seen from the studies of Hakan Tuzun and Thomas W. Malone, there seems to be little difference in what motivates based on age. We will refer to their works interchangeably as we identify the key points needed in educational games in general, and our project in particular. Even though some of the key elements presented here are not represented in our final project, we will discuss these aspects as ideal concepts.

As we review the work of Malone and Gee we initially discover something very interesting. As Gee explores the driving aspects of commercial computer games he discovers that computer games which forces learning on the player are generally more successful than those that don’t. He further implores that the learning aspects of these commercial computer games is the key factor that keep the players interested. At the same time Malone is focusing his research on educational computer games and trying to pinpoint the motivational aspects

needed in such games. This creates an interesting contradiction of sorts. If commercial computer games beget success from implementing learning aspects in their games, why is it so difficult to motivate in educational games?

From the work of Malone, Gee and Tuzun we derive some basic motivational concepts that are easy to implement, and should be apart of most educational computer games. In our project these concepts help focus our development efforts, and define some of the key elements we want in our game.

2.2.1 Challenge, Goals and Feedback

As seen in research on both commercial and educational computer games, the level of challenge, choice of goals and the feedback given as players strive to accomplish these goals, is the core to any successful computer game. Getting the level of difficulty right in computer games is not an easy task, but from the work of Malone and Gee we can derive a few simple key points that help in regards to the basic game design.

A variable level of difficulty is the most basic of these concepts. As players will always be of a varying level of skill, it is important to construct a computer game that caters to these variations. There are several ways to implement a variable level of difficulty. It can be implemented as a player choice, as a simple menu where they choose a difficulty setting they deem appropriate for themselves. This choice would then have to be directly reflected in the feedback given within the game, based on the input. With a correct choice of user defined difficulty levels, a computer game will be able to cater to a broader player base.

Another way of supplying a variable level of difficulty is to have the game automatically adjust the game difficulty based on the input given by the user. Thus trying to always keep the level of challenge at the brink of what the player is able to cope with. This is harder to implement, as rather then by pre set standards of difficulty, the computer will have to try and judge the player skill based on highly limited information about the player. It can also hinder the player in seeing progress, as the game always compensates for any increase in player skill. As this approach is both harder to implement, and possibly hurtful to the game if not implemented to perfection, we will not develop such a dynamic level of difficulty.

Another option for providing lasting gameplay at a varying level of difficulty is having

multiple level goals. This can be implemented through a scoreboard where players can strive to perfect their score, or secondary goals within the game that give some form of arbitrary reward. A scoreboard is something that can be easily implemented within most games, and something that is often overlooked. A scoreboard can easily bring replay value to most games. This is especially important in educational games, as they tend to be shorter in nature than their commercial counterparts. Other forms of multiple level goals will also be considered, such as an arbitrary reward system.

It is important through the course of any computer game, to give sufficient feedback of progress to the player. This is something that can easily be coupled with a scoreboard by giving the player access to see his current score during gameplay. Other forms of direct feedback on the player's actions should also be implemented.

If the challenge level, goal choice and distribution are correctly implemented, and coupled with a good feedback system, any game would have a good foundation for motivation. This will have to be one of our main focus points when designing the finer points of the game.

2.2.2 Curiosity and uncertainty

Another important aspect for motivating players is creating uncertainty around the outcome. A computer game quickly gets boring if the player knows beforehand that he is destined to either win or loose. There are several ways of creating this uncertainty, the perhaps simplest way, is to introduce some form of randomness within the game. With a small level of randomness, and a sufficient level of complexity within the gameplay, the player will be uncertain about the outcome of the game, while at the same time not feeling as if he or she is at the whim of a random number generator. Coupling that uncertainty with basic elements that awaken the curiosity within the player, will keep their interest in the game for a lot longer. Curiosity can be achieved by providing what Malone calls "optimal level of informational complexity". This is done by providing the player with just enough information when the game starts, to be able to understand the basics, but little enough as to leave them questioning key aspects of the game. One easy way of providing this, is by having the fantasy presented in the game be directly related to some real world setting that the player can easily identify. This will give the player certain expectations as to how things should work, but still leaving him or her in the dark about the finer points. One example of this is having the game present a fantasy of driving. The player would expect the car to turn right if he turns the steering right,

but still be curious as to where to drive.

2.2.3 Empowered learner and problem solving.

As stated by Gee, giving the player a sense of ownership and participation in the world is very important. This is what he calls Empowered learner when he is referring to educational computer games. Giving the player a sense of ownership can be done in several different ways. One of the easiest is simply to have a level of customization in the game. Customizing either the interface or the main character in some way, gives the player the notion that he has in some taken a small part in designing the game, thus giving him a sense of ownership. This can also be done by granting the player the power to change the game world based on his actions. For instance, if the player has rescued a maiden in distress, he should later on be able to see said maiden going about her life, and being generally thankful towards the player. As we noted earlier, giving the player the ability to customize the main character gives him a sense of ownership, but that is not all. It can also be used to aid the player in identifying with the character. This will give the player in a greater sense of connection to the game and game world, making the fantasy presented much more convincing.

How problems are presented and solved by the player is another aspect that is important to keep the player's attention. As Gee states, the key point here is to get the difficulty progression right. Gee calls this Well-Order Problems. This is done by supporting skill progression in the player, through linked and orderly problems. First you have to present the player with an easy task, which involves some form of skill at a basic level. The player is then given the time to master that skill, in slowly progressing difficulty, before being put to the test as he has mastered it. Only after one skill set is mastered, you present the player with a challenge based on another skill. To add complexity to this, you can later on combine several skills the player should have mastered into one challenge. When testing the player's mastery of any skill, it is important to give them a sense of progression, should they fail at the challenge, this is what Gee calls “Pleasantly frustrating”. Players are presented with a hard task, but they should always believe that it is doable, and that they are progressing towards its completion.

Gee also introduces the notions of fish tanking and sandboxing when it comes to game design. Fish tanking, is a design form, where you introduce a simplified version of the game

environment, with the notion that it is easier for the player to learn certain aspects of the game, if not presented with too much information at any one time. This makes fish tanking an effective tool for tutorials and introduction. Sandboxing is a similar concept, made to make the introductory phase easier to cope with. In a sandbox environment the player is presented with a risk free version of the game in general, or a more specific task within the game. This makes it a lot easier for the player to experiment to find solutions as there is no punishment for failure.

When the player is left to improve his skills, either through sandboxing, fish tanking, well-ordered problems, or similar means, it is important to always give the skill progression purpose. Skill progression in itself can feel meaningless if the player doesn't have sufficient incentive to do so. This can easily be done through setting goals related to the skill improvement, as discussed earlier. Another way of doing this is by giving the player a sense that the entire game world is somehow related. For any task or challenge that is completed, there should be visible changes within the world which directly affects the gameplay.

2.3 Summary

Based on our research of earlier work we have found that motivation has to be the main focus when developing any educational game. We have also identified some of the key aspects that provide motivation in computer games: Challenge, Goal, Feedback, Curiosity and Uncertainty. We explored the benefits of these elements within games, and how the combination of them gives the players a sense of accomplishment. Our game has to be centred on these motivational aspects in order for us to test if educational computer games can be used to motivate students at a university level. We hope to contribute to the research done by Gee and Prensky by providing practical testing of an educational game.

Chapter 3
Designing an educational game

In this chapter we will look into what makes an educational game. We will present different game types, and discuss the benefits and problems of each. Based on the discussion around the different types, and linked to the research we did in chapter two, we will present our choice of game type.

3.1 What is a game?

As stated by Gee (", 2005), in order to make an educational computer game, one must first look into how games made for entertainment purposes are designed. First we have to identify what a game is. Award winning game designer Wolfgang Kramer («Kramer», 2008) explores the definition of a game in his article, «*What Is a Game?*» (Kramer, 2000). Kramer seeks to narrow the definition down from former definitions that described games as an art form, or a natural phenomenon. He seeks to define what he calls "games with rules", which is to say anything we now label as games. Kramer seeks to define the different criteria that make up a game, and we will derive those criteria from his article.

3.1.1 Components and Rules

All games must have rules to which the players have to abide by. The game rules clearly define what a player can and can not do when playing a game. Kramer stresses that a game must also have components aswell as rules. As he describes it, the components are the hardware of a game while the rules are the software. Components can consist of the game board or game pieces, while the rules govern how these pieces can move across the board. Components and rules can exist separately from each other, but do not make up a game on their own, as explained by Kramer's example of archaeological findings of ancient game boards and pieces. Without the rules adhering to those components, it is impossible to know

how the ancient game was played. Components and rules can be interchanges. One can use the same set of components with different sets of rules, and one can use the same rules with different sets of components, but a game always needs both to be playable.

Components and rules define the game, anything that is described by the rules, is a part of the game, and anything not described by the rules does not belong to the game. Thus these criteria define the border that makes up the game. Even though rules are forced upon players, partaking in the game needs to be voluntary for it to be called a game.

3.1.2 Goal

Every game must have a goal, something that defines a successful outcome. Kramer separates the definitions of the goal into two separate definition. One is the victory conditions or the requirements, and the other is strategy needed to win the game. The goal needs to be something that is measurable, and ideally relatively easy to measure for the player. Even though there are an enormous amount of different games, there are a relatively small number of different goal types. Goal types usually consist of simple quantifiable end conditions and are either defined as a set amount or relative to other players. Examples include; make more money that the opposing player, or make one million dollars.

3.1.3 Chance

Another aspect of games that make them differ from movies and books is uncertainty of outcome. You can read a book or watch a movie multiple times, and the outcome will always be the same. The main reason games differ from other media in this regard, is that they have some element of chance. Each time you play a game the outcome will be different. Chance in games can come from four different sources according to Kramer.

- with a random generator (e.g. dice)
- with different start-up situations (e.g. dealing cards)
- with incomplete information (e.g. moving at the same time, unknown strategy of your fellow players)
- with a very high number of move options(e.g. chess)

Chance is what makes games enjoyable to play over and over without loosing their appeal.

Some games rely more on the element of chance, like Yatzy while games like chess has no real element of chance, just an obscene amount of possible moves, making it highly unlikely to see the same play more than once. Chance is essential for a games replay value.

This category is one that narrows Kramer's definition of "games with rules" as he want to exclude puzzles, quizlets, and brain teasers, which he claims looses their appeal as soon as they have been solved.

3.1.4 Competition

Competition is also a must in all games according to Kramer. Competition is usually against other players but can also be against some predetermined goal in which the players compete against the game system. For a game to have competition it needs a system for which the end results can be compared. This usually comes naturally from the way the goal is determined. Goals are usually set as either the defeat of the opponents or reaching some goal that is quantifiable in a score. This end score can be used to compare achievement in the game, and gives competition.

3.1.5 Other Criteria

Kramer goes on to describe game criteria for games that does not fall under his category of "games with rules" These criteria will only be discussed briefly as they have less to do with our project of creating an educational computer game.

Common experience

A lot of games have the quality of bringing people together, working on a common task, regardless of age, gender and culture. Single player games (e.g. solitaire) are excluded from these criteria. Kramer also mentions computer games as excluded from giving a common experience, but forgets online games like *World of Warcraft* (World of Warcraft, 2008) that perhaps more than any other type of game brings people together.

Equality

One of the unique qualities contained in most games is how they put players on an equal playing field when they start. As Kramer asks: "Where else in this world does absolute equality exist?" (Kramer, 2000) This is one of the factors that people of different background

and qualifications come together while playing games. Children get the opportunity to compete against their parents in something and actually have a chance of winning.

Freedom

Playing games are a matter of choice. People are free to choose to play games, and this gives games a unique feeling of freedom while playing.

Activity

One of the things that makes games differ from other media like books and movies is that it forces activity. This in itself forces people to think and reflect on what they are doing, thus making games an ideal backdrop for learning.

Game World

Most games give the players a sense of escape from reality. When all actions taken in a game is separated from real like consequence the player forget worldly troubles, at least for a while. This is not true of games Kramer labels as "reality games" games that greatly influence real life situations, usually games of chance like poker etc.

3.2 What makes people play games

As we saw in the work from Gee (", 2005), the learning aspect of normal computer games is one of the key factors that actually make people play games. Most people like to be challenged, at least in a safe environment. To then overcome these challenges brings a feeling of self worth and accomplishment. Good computer games have incorporated learning in a way that makes it biologically motivating, according to Gee. This is incorporated in games through the elements described in his text. We reiterate the main points derived from his text.

Empowered learner:

- Co-design
- Customize
- Identity
- Manipulation

Problem solving:

- Well-Order Problems

- Pleasantly Frustrating
- Cycles of Expertise
- Information “On Demand” and “Just in Time”
- Fish tanks
- Sandboxes
- Skills and Strategies

Understanding:

- System Thinking
- Meaning as action image

As we are designing an educational computer game, it is important for us to incorporate these and other forms of motivation. Unless students are motivated to play the game, they will not learn anything. By ensuring that the motivational aspects are in place, we also ensure that we have created a platform suitable for learning. We can then add specific learning into our game to tailor the learning environment towards our goal of teaching system development.

One of the additional characteristics mentioned by Marc Prensky (", 2002) in his paper "In Educational Games, Complexity Matters" (Prenzky, 2005) is how a sense of accomplishment is one of the main reasons why people play computer games. As one progresses within a game one gets the same sense of accomplishment as one can get from progressing in sports, hobbies or work. As this is one of the major driving forces in real life it is only natural that this is one of the major factors that motivate us in computer games as well. Prensky's notion of accomplishment from playing computer games is directly connected to Gee's notion of computer games as learning machines. Gee convincingly argues that games that teach us something (learning machines) are more motivating than games that fail to do so. Learning a new skill or gaining knowledge is in itself accomplishments, and this directly connects their two papers. Prensky focuses on game progression, through complex games where the avatars evolve through levels and learn new skills in the game. This is directly connected to Gee's "Empowered Learner" because the player has to learn how to use these new skills attained by his avatar.

This duality of progress and learning is crucial when developing any computer game, whether it is an educational one or not. We need to present the players with an evolving game that

changes as they progress. This forces the players to learn new skills as the game develops as well as giving them a sense of accomplishment.

3.3 Important aspects of an educational game

Earlier we have stated the importance of motivation in games, and how especially important it is to educational computer games. With this in mind we try to design for fun and appeal firstly, but we must not forget the educational part. This is where educational computer games differ from their more commercial counterparts. In order to ensure that learning is facilitated after the motivational aspects are in place, educational games need to mimic the teacher-student relationship. This is a relationship where learning not only comes in the form of an information stream from teacher to student, but ideally works as a conversation where the student gets constant feedback based on performance. As discussed in Ekaterina Vasilyevas' paper, Towards Personalized Feedback In Educational Computer Games For Children(Vasilyeva, 2007), feedback is also one of the key features to a successful educational computer game.

In her paper she defines the different forms of feedback in computer games, and more importantly their function. Some of the functions performed by feedback are; informing the user about his performance, keeping attention and motivating. However, the most important part of feedback is probably rewarding the player based on performance. Visual or audio feedback can be used as form of reward for the player, and does not need to affect the gameplay. Examples of this include visual upgrades for the avatar, a gallery of trophies or a simple informational text describing the player's success. Other forms of rewarding feedback may directly affect the further gameplay, such as additional levels for the avatar or in game currency.

In educational computer games, feedback performs another important aspect, supplementing the learning process. By providing sufficient and appropriate feedback when a player fails a task in the game, the player can better learn from their mistakes. This is one of the main benefits of educational computer games versus ordinary educational environments. Educational computer games can facilitate direct and immediate feedback to the player during the learning process. In schools and universities, students usually have to wait for test results or compete for the attention of the teacher with all the other students. This somewhat limits

the amount of feedback students get during traditional education.

The feedback in educational computer games needs to be detailed enough to assimilate the feedback a teacher could give in the same situation. Without this, the games are degraded into tools only useful for brushing up on previous knowledge or testing this knowledge.

3.4 Different types of educational games

One of our goals with this game is to make the students have fun while learning system development. We need to make sure the student are entertained while playing the game, and at the same time give the student enough system development related task to ensure that they actually learn system development. To do so in the best possible way, we need to look more into what kind of game that could suit our task. In the following chapter we have listed different type of game concepts that could suit our task, and written down positive and negative sides of each concept. To come up with the different concepts we have used our own gaming experience, and looked at different game concepts used in well known and popular games.

3.4.1 Multi user Dungeon (MuD) game

It is possible to make a multiplayer text based online game. In this game one will take the concept of old MUD based games and adapt it to our educational purpose. A MuD game is usually text based. The player takes the role of a character that move around in a virtual world that is described through text. The players interact with each other and with the surroundings by entering commands in a command window. Because of the nature of a MuD game, the gameplay, fun and nostalgia (for those who played the old MuD games) will probably come first, and the educational part of the game will come as secondary information.

Example:
You are a mail man working at the bottom floor of a 15-story high building, and have to solve different types of puzzles to work your way up through the building. As the game progresses you gain different work experience, and get hired in more attractive jobs (mostly computer related in our case, as we are trying to educate people on system development). A typical scenario for a MuD game could be: “You have entered the 10th floor where you are a newly hired project coordinator. When you exit the elevator you are presented to your new project

group. They explain how they are in the middle of a large dispute on how to solve their ongoing project." You then have to move your character around the floor, interact with people, and help the group solve the problem. If you solve the problem in a satisfactory way, you can move on to the next floor, and maybe even get a promotion.

Benefits

- Possible to reuse old MUD-engines in the development of the game
- Nostalgic to play a MUD game for many people
- Possible to implement multiplayer
- Fun to play with others.

Drawbacks

- We doubt that text based games have the same appeal as graphical games
- Could prove hard to implement good multiplayer
- Hard to implement system-development related tasks without having graphical interface

3.4.2 Football manager – Window based single player JAVA

The idea here is to do a remake of the popular game type Football Manager ("Football Manager", 2007), but instead of letting the player control a football team, the player controls a system development company. In this game the player takes the role of a project leader/project employee/company leader, and is given different system development tasks. The player is given different problems to solve, for instance how to administer resources, development methods etc.

Example:

The player is a new employee in a System development company, and is given the task to manage a project. The player hire/fire employees based on different computer skills, and build up a project group (One can clearly compare this to how a football manager manage a team). The player is given a time line for the project, and during the project the player manages the workers, work tasks and problems that occur during the project. Maybe the workers come to the project leader (the player) and ask for advice on how to make a grouped entity relationship model, and the player has to use an in-game drawing tool to make this. After a successful project, the player gets access to larger and more advanced projects to lead.

Benefits

- Good possibilities to implement system development learning.
- Possible to reuse open source football manager games. For instance Bygfoot Football Manager (“Bygfoot”, 2007)
- Relatively easy to implement the basic interface and unlimited possibilities to further implement more advanced features (for instance drawing boards).
- Large possibilities to make a game that covers the task at hand.
- Should be possible to implement multiplayer feature.

Drawbacks

- Football Manager is a popular game because one manages a known football team with known players. Would a Football Manager game without the football be the same success?
- Much work in implementing good feedback on advanced tasks.
- Much work in implementing large and difficult problems to present to the player.
- This solution could prove to be a very large task to implement, and would most probably just be a proof of concept solution.
- Boring to play alone?

3.4.3 Single player career-game based on the game, Jones in the Fast Lane

Turn based single player game where the player takes the role of an up and coming person in the system development business. The player got the opportunity to evolve through education and work, and use the money he earns to improve his status by buying a nice house, furniture, car and clothes. The game itself is very much like a board game. The interface shows a picture of a city the player can move around in. In this game the clue is to balance the three main issues in the game, namely work, education and money (status). This idea is based on the old classic Jones in the Fast Lane (“Jones in the Fast Lane”, 2002).

To implement system development learning in this game, we provide good system development literature in the in-game school, the in-game work gives system development related tasks through different types of mini games, and the status part of the game is there to have a ranking system that encourage the player to improve his character.

Example

The player in this game starts out with nothing. He gets a story on how he has just been kicked out from his parents' house, and need to make his own living. In order to get an education, the player has to go to the bank and get a loan in order to start his education. Once he has taken the most basic courses at the in-game school, he can apply to a low income job. While working, and solving different system development tasks, he earns money to improve his education and after a while get a better job.

Benefits

- Easy to implement a good interface
- Encouraging gameplay
- Easy to implement turn based multiplayer
- Can have a nice appeal with 3d pictures in the user interface
- Easy to link education and system development learning

Drawbacks

- Hard to implement simultaneous multiplayer
- Time consuming to implement 3d pictures to give the game a good "feel"?

3.4.4 Web portal for group assignments

The idea here is to make a server/client program to publish and work with group assignments. Through the portal one can solve and hand in group assignments, and get access to other peoples assignments to get a view at what others are doing, and to correct them in order to help others out. The content of "the game" will be given as assignments formulated by an administrator (typically a group teacher), and one is given a time limit when the assignment has to be handed in. It is also possible to implement a ranking system that encourages the students to work harder and be more thorough on their deliverables.

Benefits

- If this is implemented right, this is a "game" that the INF1050 course at UiO would want to use.
- Possible to give hard assignments without having to program them into a game.
- The content of the game would be updated frequently as the administrators add content all the time.

Drawbacks

- This is not a game!
- Similar software already exist (“Fronter”, 2008)

3.4.5 Evolving puzzle game

This game type is based around the educational content. It works as a series of assignments that the player has to answer using knowledge based on the theme that the game is supposed to teach. Which assignment the players are presented is largely based what they answered on previous assignments. This game type does not have any notion of a game world except from textual descriptions that can be given with the assignments. There is little use for an avatar, and the player's connection with the game would most likely only come from referring to their chosen name in the assignments. The main benefit of the Evolving puzzle game is its keen focus on the educational content. Every part of the game would be focused around using extrinsic knowledge to solve puzzles.

Benefits

- Focus on educational content
- easy to implement

Drawbacks

- few game elements
- might become repetitive without a large amount of different assignment types
- low immersion level
- would it be entertaining?

3.5 What type of game we decided on and why

In order to decide on what type of game we would develop, we made a list of attributes that we meant were important to educational games. We found these attributes by looking at the research others had done and from our own experience with games.

Entertaining

One of our plans for this game is to get students to test it in their spare time. In order to get as many as possible to spend the time on actually doing so, we felt that the game needed to be entertaining. This entertainment could be implemented in several ways dependent on what kind of game we choose. In some games the entertainment could be implemented through exciting and humoristic content, while in others through stimulating the player's sense of accomplishment.

Educational

It is important that the game type we choose is capable of adapting educational content into the game in a good way. What we especially look at when choosing game type is how advanced system development questions we can integrated in the content. We want it to be easy to implement advanced questions, but at the same time it must not be too time consuming in doing so.

Immersive

Once we got the players to play the game, it is important that we hold their attention. Therefore our game type needs to be immersive. In order to achieve this we will look at several aspects. We need the game to be capable giving “optimal level of informational complexity” as Malone calls it (Malone, 1980). The players' needs their curiosity stimulated, so they are wondering about what is to come next in the game. From our own experience we know that games can quickly become boring if they are too predictable. If the game type we chose is capable of immersing the players into the gameplay, and at the same time be entertaining we believe that we got the basis for keeping the players attention.

Appeal

In order to get the players to try the game in the first place, we believe that it needs a certain appeal. The players are motivated through the fact that they can learn new knowledge, but we believe that a good appeal to the game would further motivate them to try it. Therefore we decided that the appeal of the game should be taken into consideration when choosing game type. As the same game can have different appeal from person to person, we have used common sense and discussed with each other when we rated the game types on this attribute.

Is a game

For our project it is crucial that the players feel that they are actually playing a game. We are doing research on educational games, and if the game type we chose could be looked upon as more of a program to stimulate regular school assignments, we would fail.

Ease of implementation

For our research we have limited time on developing the actual game. We need the game type we chose to be advanced enough for us to be able to test our thesis. At the same time we can not allow ourselves to spend too much time on writing the actual game code. When deciding on game type we therefore need to consider how advanced and time consuming the different game types is to implement. We did this by considering different implementation issues in the different game types up against our own programming experience and how advanced and time consuming we thought them to be.

To get an overview, we rated the different game types up against our criteria in the following table:

(Attribute: 1 = poor, 2 = average, 3 = good. Ease of implementation: 1 = hard, 2 = average, 3 = easy)

Attribute	Multi user dungeon	Football manager	Jones in the Fast Lane	Web-portal for group assignments	Evolving puzzle game
Entertaining	2	3	3	1	2
Educational	1	2	2	3	3
Immersive	1	2	3	2	2
Appeal	1	2	3	2	2
Is a game	2	3	3	1	2
Ease of implementation					
Educational content	1	2	2	3	3
Goals	1	1	2	3	1
Graphical user interface	1	2	2	3	2
Educational feedback	3	2	3	1	3
Interaction with the game world	2	2	2	1	2
User customization	1	2	2	2	1
Multiplayer possibilities	3	2	2	1	1
Sum	19	25	29	23	24

Table 1: Game concepts vs. implementational aspects

In the table above we notice that the Multi user dungeon game fails our criteria on several points. The game type is not especially suited to use in an educational setting. It would be relatively easy to develop such a game, but would be very hard to add good graphical educational content to a text based game. A solution to this could be to implement graphical user interface with avatars that moved around in a game environment, but we believe this to be quite hard in our time frame. We decided to drop the multi user dungeon game type.

We quickly decided that the Web-portal for group assignments solution would not meet our demands for the game we wanted to develop. When reviewing Kramer's article on what a game is, one quickly sees that this solution is not a game. At the same time, we know that these kinds of Web-portals designed to assist the students in their deliveries and such already exist.

A very interesting possibility for game type is the Evolving puzzle game. In this game type the educational content is the central theme, and the gameplay itself is secondary. This game type is excellently suited for making educational questions and giving good feedback. The issue with this game type is to get the player to feel that it is a game and not educational questions given out in a chain. As we can see from the table it could be hard to implement goals for the player in this game. Goals such as "answer these four questions correctly in order to advance" is rather easy to implement, but to add a story to the whole game could prove difficult. This game could make the game content very repetitive and few new elements other than new questions will arise while advancing in the gameplay. If the player is presented with nothing else than new questions all the time it could become predicting and boring to play and the player would loose interest.

The Football manager type of game is a game type we instantly liked when we came up with the idea. It is in our belief that the Football Manager concept has a very good potential for learning the players system development, but we also think that it would loose much of its appeal when the content of the game is about system development and not well known football teams and players. Then again, when the educational content in the game is about something the player is interested in learning it might work out anyway. As we see from the table the game type scores high on all of our chosen attributes. Our only concern with this game type is that it could be very complex to develop in a way that the gameplay feels natural and immersive.

Jones in the Fast Lane is also a game type that we instantly liked a lot. It scores high on our criteria and is a game we have both played, and liked, in our youth. We especially liked how it is possible to separate the educational content and gameplay the way it does. This makes it easy to develop a platform for our game and add different educational content based on difficulty level and subject one want to teach the player. A game cycle in this game type is very quick, and one develops the avatar in the game quickly. This gives the player a feeling of ownership over the avatar and stimulates the interest to play the game further to see how far it is possible to develop the avatar. From our own experiences with the original version of this game we remember how it was very good to stimulate the curiosity of the player. As the turns in the game are quick, there is almost always something new to in the game. We believe that short intervals between new information stimulate the curiosity and thus make the game more immersive. When it comes to ease of implementation this game type is also quite nice. Its layout is like a board game and therefore there should be no advanced UI programming. It could be difficult to implement advanced question types, but as the educational content is separated from the gameplay we can easily control the time used on developing them.

After reviewing the different game types we found out that it was three concepts that could suit our project, namely Football manager, Jones in the Fast Lane and Evolving puzzle game. As we felt that football manager and Jones in the Fast Lane are more typical computer games than the evolving puzzle game, we decided to drop the evolving puzzle game. For our project we need to make a game that give the player the feeling of playing a typical game, and we didn't feel we could get that out of the evolving puzzle game.

When comparing Jones in the fast lane and Football manager in our table list we see that they only differ in two attributes, immersive and goals. The typical gameplay in a manager game is that the player is presented with a lot of information, and needs to take actions based on the facts and what the player believes is right. We believe that for a student trying to learn something new it could be a lot of information, and difficult to grasp everything in the beginning. It could be a nice way to learn, but it could also be very hard to implement for us programmers. Setting good and clear goals in a game where there is a lot of different information is difficult. At the same time, we know from experience that programming a game where very many things are dependent on other elements in the game could be very difficult, and very hard to balance.

We made our choice based upon what game concept the players of the game would like best. Fun and play is a very motivating fact when it comes to games, and we believe that this is the major issue for us. We need our game to be fun, so people keep on playing the game. If we made a game that was purely educational and no fun, people wouldn't play it, and as a direct consequence of that, our project would fail. Further we want to develop a game the player easily understand and do not need to spend a lot of time just to understand the gameplay. As Jones in the Fast Lane game type got all this, we decided to go with that game type.

3.6 Summary

In this chapter we defined that games are comprised of components and rules, and applied this knowledge to studies on computer games. We looked into the elements that make up a computer game, and the importance of these elements in educational computer games. These elements closely resembled the motivational aspects we had found in our research of previous work.

With this knowledge we went through different forms of educational game implementations and identified their weaknesses and strong points related to our research. Having done this, we decided on an implementation based on the old classic Jones in the Fast Lane. We chose this implementation because the criteria we had researched could easily be implemented within this game.

Chapter 4
The educational content of our game

In this chapter we will look more into the system development content we would like to add to our game. As the goal of our project is to make an educational system development game, we need to add the appropriate system development knowledge into the game. We started out by looking at the curriculum of different courses that is taught at the University of Oslo and asked the informatics professors at campus what they thought would be appropriate content to add into the game. We quickly found out that the curriculum of a course taught at UiO, namely INF1050, covers the main aspects of System development. Here is an excerpt from the INF1050 homepage:

> **Course content**
>
> Development of information systems. Methods, techniques, computer based development tools, technological platforms. Influence of systems on their environments – individuals, organisations and society
>
> **Learning outcomes**
>
> After the course, students should understand what it takes to develop an information system: Determination of requirements, stating the various constraints imposed on the development, efficient development and deployment, and managing the system development process.

(«INF1050», 2008)

As we can see from the course content and learning outcomes stated above, this course covers the main topics we want to include in our game. Therefore we chose to use the teaching material of INF1050 as starting point for the system development content in our game. The

reason for choosing a course lectured at UiO as a starting point is that we will have access to students that can help us test the game. At the same time, if the game is a success, the students can benefit from learning while playing the game, and thus help them in the course.

We sent some emails around to the group teachers of INF1050 and asked them what the students had problems learning and what the group teachers thought was the central points in the curriculum of the course. Through the communication with the group teachers, and from our own experience in attending the course, we found out that several of the students lack some prerequisite knowledge that is central in the course. This includes SQL, PHP and programming in general.

Late through our work on this thesis, the INF1050 curriculum went through a complete overhaul, and a lot of what we had identified as elements we wanted to test the users in was removed. We had to do another analysis of the course to figure out what to include in our tests. This proved to be somewhat difficult as the curriculum had not been finalized as we were about to perform our tests. We decided to limit our testing to some central points which we knew were going to be in the curriculum, and not test the students in any of topics that there was uncertainty about whether or not would be included in the curriculum. We will discuss the ramifications of this change later in the chapter.

4.1 What is covered in the curriculum of INF1050

Here we have extracted the main points covered in the INF1050 curriculum. This is taken from the book "*Systemutvikling - fra kjernen og ut, fra skallet og inn*" (Skagestein, 2005), lecture slides used in the course (INF1050, 2008) and feedback from the group teachers. In this chapter we will present several terms from the INF1050 course, but we will not go into them in detail, as that would result in us reciting large portions of the book and slides. The terms are rather presented here to explain what aspects of the book we choose to include in the educational part of our game. The target group of our game is expected to know most of the terminology by attending the course.

In 2008 the course stopped using "Systemutvikling - fra kjernen og ut, fra skallet og inn", and rather chose two new books. These two books, "*Requirements Analysis and System Design*" (Maciaszek, 2007) and "*Grunnleggende systemutvikling*" (Gurholdt and Hasle, 2003), cover

the same topics as the book used in 2007. As we had already started to work on the 2007 curriculum when they changed books and curriculum, we decided to stick with the content we had made, and rather leave out questions we had made that would not suit the new curriculum.

Top-down or bottom-up approach to system development

As the title of the book suggests, one of the main points of the book, is to introduce the reader to two completely different ways of looking at system development: The "Top-down" and "Bottom-Up" approach. The students are expected to know the key differences in the two approaches and be able to outline the idea behind them. Further, they should be able to know when one is preferred above the other and explain why.

System development strategies

The students learn about different system development strategies, and are expected to be able to describe the idea behind them, identify them, and explain their benefits and downsides. Students should also understand why and how to adapt these methods to different project demands. The strategies should also be related to the two main approaches to system development. Given a project description they should be able to choose a development strategy, explain why the elements contain in the chosen strategy is suitable for the given project, make necessary adjustments, and affirm their choice by connecting it to the theory related to the main approaches of system development.

Development platforms and software

The students are expected to learn the basics about different development platforms and have knowledge about how a computer is working. The book covers the basics on how programming works and how one can use different development software to aid the programming process. Further the students are introduced to the benefits and downsides of different ways to organize the architecture in a system development project.

Relational database

The book covers the basic concepts behind relational databases and their use. The students are expected to learn how to set up and interact with a simple database using SQL commands. SQL is not directly thought in the lectures, but students are given a basic introduction to SQL in the study groups. SQL queries towards a given database to extract information are an

important part of the course.

Data models

The curriculum gives an overview of how to describe the real world with data models. These models can then later be used as a basis for setting up relational databases for storing information about real life situations. The models presented in the book follow the standards of UML diagrams. The students are expected to be able to create data models, extract a relational database from it, and also improve on existing data models.

Normalization and redundancy

The students learn about different levels of normalization in regards to relational databases. They are expected to be able to change simple database examples to different forms of normalization, discuss preferred levels of normalization for different examples and identify normalization levels in database examples. They should also be able to identify the need for normalization by pinpointing redundancy in either relational databases or data models.

Constraints

The book presents the different types of constraints that are normally used in relational databases and their use is explained. The students are expected to know the different types and why they are useful. They should also be able to identify and add necessary constraints to simple data bases.

Data representation

The book covers the various data representations that are used in relational databases, how they are stored on the computer, what computations that can be done on them, and their different uses. The students are expected to be able too choose appropriate data representations for information presented in a project. They should be able to discuss the benefits of using one form of data representation rather than another in a given example. The book also covers identifiers and various levels of information carried in data representations.

Designing user interfaces

A portion of the course is dedicated to designing user smart user interfaces, and several common methods for designing user interfaces are presented. The students are expected to be able to design user interfaces through the study and creation of Use-Case diagrams. From

these diagrams the students need to be able to extract system requirements for a given project.

Designing top-down

The student learn how to develop a system in a top-down method. When finished with the use-case models and the different user interfaces in a system, one needs to look more into the actual core of the new system. There are several ways of developing the core of the system, but the author of the book is mainly focusing on object oriented architecture. The students are introduced to object oriented philosophy and learn how to define the different objects by using methods like CRC-cards and UML sequence diagrams.

Evaluating information systems

When the system development has started or one is finished with a new product, several new aspects come into play. The students learn how to consider if a system that is under development will work as intended when it is finished and how to find out if a new system really is worthwhile to use. In the light of this the students also learn different data collecting techniques and knowledge about the different forms of data evaluation that is used.

Knowledge about different issues involved when making a contract between the client and the developer

The students learn three different forms of contracts between the client and the developer. The main theme here is how to make a contract that both the client and the developer can agree upon and relate to. A system development project is often subject to changes and the book discusses how these changes are to be handled by writing different forms of contracts. Another important aspect is the problem of how to specify the actual content of the system that is under development. The developer and client often see things from different perspectives when it comes to the actual development, price and delivery time is an uncertainty and how this should be included in a contract is looked upon.

Price and time estimation

Early in a system development process, one often need to consider if it is cost- and time effective to develop a new system or not. The students learn different principles for estimation of workload and locate uncertainties when estimating in a development project. This again is linked up against contracts between a client that orders a system developed and the developers, and further linked to different ways of agreeing upon the cost of the project.

Law and ethics

An important aspect of System Development is Law and ethics. The student is required to understand how system development is affected by both Norwegian law and ethical rules. Especially important is the laws concerning storage of personal information. The goal here is to teach the students the basics of the laws they need to be aware of. The book does not go in depth on the juridical questions.

4.2 Different implementations

In this chapter we have looked at different types of mini games, and made some suggestions on how we can implement them into the game. Every mini game also has several levels of difficulty. All of the difficulty levels require different types of implementation into the game. We also need to consider how time consuming it is to implement the different types of mini games. The content of the mini games listed below are just examples on how they can look like and what they can teach the player.

In order to keep the attention of the students playing our game the difficulty level on the questions in our game should start easy and be harder as one advance in the game. At the same time we were very careful of adding very difficult content as the player is required to answer all questions correctly in order to advance in the game. If the questions were too difficult this could result in the player quitting the game, and our goal of testing if educational games are a nice way to teach university students would suffer.

4.2.1 Multiple choice

In this mini game the player is presented with a text question and is given a set of potential right answers to chose between. The player then have to chose an answer from the list and the answer is then checked up against the correct answer. Even though this is a very simple form of questioning it is probably the most versatile question type we can implement, because the variety of questions that can be asked in this kind of mini game is very large. To extend this question type it is also possible to show an image along with the question text.

The best way to implement this into the game is probably to pop up a new window that shows the question, possibly an image, and a radio button for each answer choice. After the player has made a choice he/she can press an "OK" button also showed in the window to confirm the answer.

QUESTION:

You have developed an information system that shows what students attend which courses at the University of Oslo. The system is open for insight to everyone on the World Wide Web. Is this a system you can use legally?

1. Yes
2. Yes, but I need to inform Datatilsynet 30 days prior to start using it.
3. No, I first need an approval from every student that is listed in the system
4. No, I am not allowed to post this information on the World Wide Web.

Illustration 1: Example of mutliple choice mini game

4.2.2 Click on image

The idea of this mini game is to challenge the player to find a specific point in an image. The idea is to let the player use the mouse cursor to locate the right coordinate in the picture and click. By using this kind of questions one force the player to analyze for instance a diagram given by a picture, and find out the correct answer according to the question given. This ensures that the players have to think for themselves rather than pick out the correct answer from a suggested list such as in the multiple choice question type.

The best way to implement this question type into the game is probably to show the picture in a window and pick out the coordinates the user chose to click on. Then compare the user generated coordinate up against the right answer. To ease the work for the person creating the questions we will probably have to make a nice graphical user interface for the actual making of the question. This will make sure that the person creating the question can find the right coordinates for the answer easily.

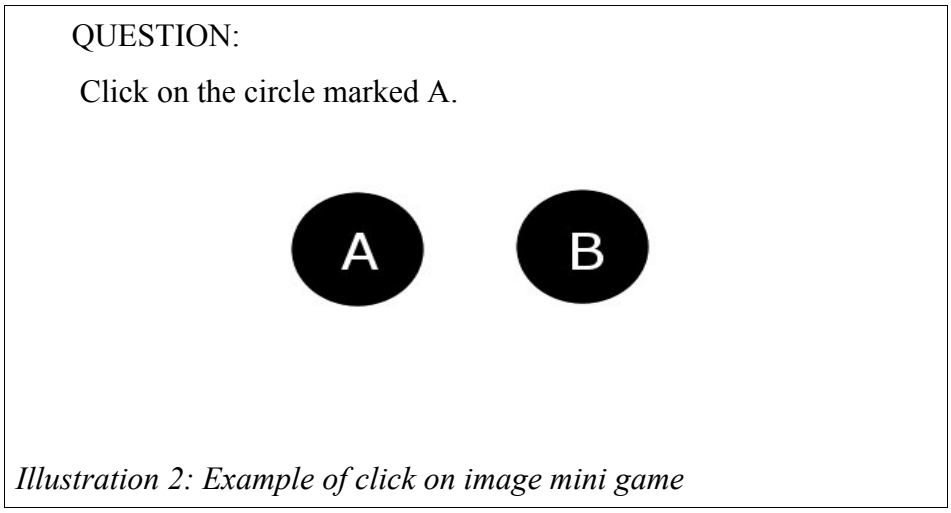


Illustration 2: Example of click on image mini game

4.2.3 Draw on image

Draw On Image is a more advanced version of Click On Image. The player is given a question and a corresponding image, but is asked to draw the answer on the image. The player then uses the computer mouse to draw a line on the image. This question type is used to test for more advanced diagram understanding. The player can be asked to draw the missing parts of a diagram, connect objects with a certain similarity, or draw in certain elements like constraints on a database diagram. This question type should not be too hard to implement in the game as there already is a paint component in the existing Java library.

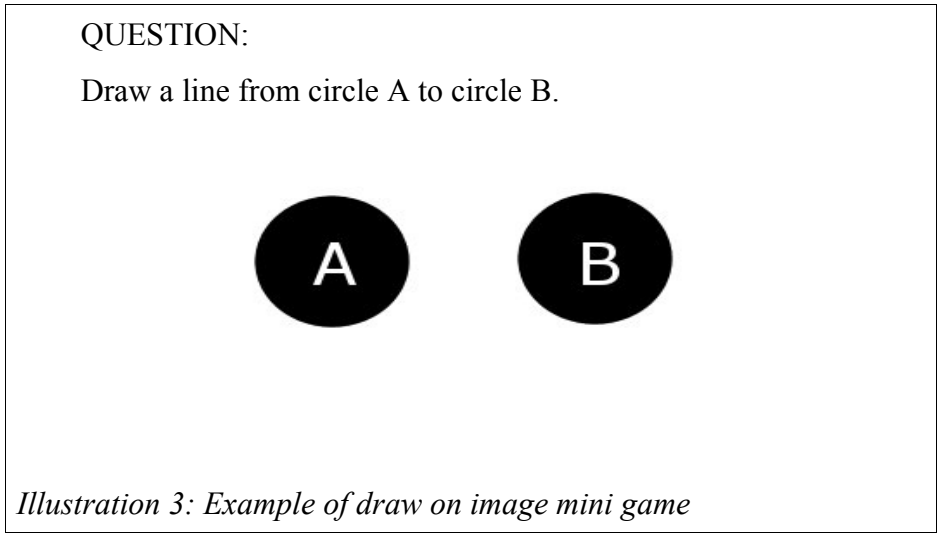


Illustration 3: Example of draw on image mini game

4.2.4 SQL question

The idea of this mini game is to make the player write in a correct SQL-query according to a given question or image. The player is shown a relational database, and is given a question on how to extract certain information from the database by writing a SQL query. The player then writes the SQL query in a text field and the query is checked up against the correct answer.

A possible difficulty in implementation might arise here. As known there are often several ways of writing a SQL query that extract the correct information. A solution to this could be to use a SQL parser that checks the answer up against database text files. A free open source Java library one could use here is *Zql: a Java SQL parser*. («ZQL», 1997)

Another solution is to actually link the game up to a database so the game can run the SQL-query up against the database and compare the results with the correct answer.

There could also be quite difficult to give good feedback if the player writes the SQL-query wrong. A possibility here is to use the standard SQL feedback one usually gets from databases.

The goal of this mini game is to learn the player more about how SQL works. Another important aspect of this mini game is that it gives a hands-on experience on writing and understanding SQL. This mini game will also teach the student to read and subtract information from a relational database. This mini game could easily be modified to give the player PHP problems and through that teach the student PHP.

Relational database:

MOVIES

MovieID	MoveTitle	ProductionYear	Company
01	Star Wars	1977	20th century fox
02	Finding Nemo	2003	Walt Disney Pictures

Question:

Write a SQL-query that lists the movies produced after the year 2000.

Illustration 4: Example of SQL mini game

4.2.5 Number compare

The idea of this mini game is to let the player write in numbers manually into the answer field, and compare the numbers with the correct answer. This mini game could fast prove to be an easy "fill out the X's and Y's in a formula and compute the answer" math type of game. The fun and educational part of this mini game is rather narrow, if it is purely a math formula that needs to be solved by the help of given attributes. To make this question type more educational we could add an image to the question which the player would have to analyze before answering.

The implementation of this mini game should be rather easy. One probably just needs to parse the user-input numbers and compare them up against the answer.

QUESTION:

The numbers in the picture indicate the time used to go from one node to another. What is the shortest time to use if you want to go from A - F?

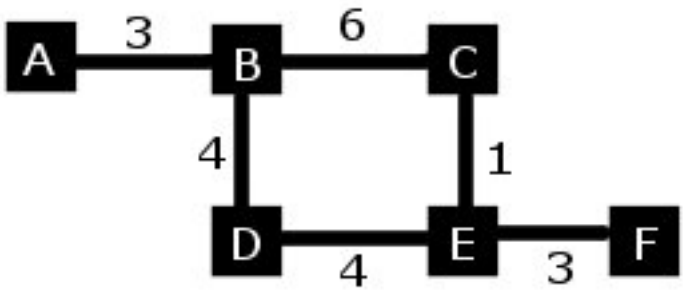


Illustration 5: Example of number compare mini game

4.2.6 Text parsing

This is an advanced version of the number compare mini game. The idea here is to let the player write in text manually in a text box, then use regex to correct the answer. The drawback of this mini game is that it could be hard to implement in a good way. One would have to use regex and correct the answer up against some predefined correct answer-words. On the positive side it is a mini game that one could use to test/teach the player in many different forms of knowledge as it forces the player to think for him/herself.

QUESTION:

What are the benefits of using the spiral method?

Answer: [*Text field*]

[OK-button]

Illustration 6: Example of text parsing mini game

4.2.7 Drag and drop

The idea of the drag and drop mini game is to let the player solve a question by dragging movable objects around in a window. A typical problem could be to presented the player with several drag-able boxes with words on them, and ask the player to arrange them in the correct order according to the given question. This mini game could be quite hard to implement, but could prove to be a very good question form. It force the player to analyze and think, and could prove to be very similar to the question types the players might get in an exam situation.

QUESTION:

Use the mouse to drag the vowels into the square and the consonants into the circle.

Illustration 7: Example of drag and drop mini game

4.3 What parts of the curriculum is best suited to implement in the game within our time-frame

Since our thesis is mainly focused on the motivational value of computer games at a university level, we chose to add appropriate educational content, but at the same time not spend too much time doing so. As have limited time on our project, we chose to add the educational content that is most relevant and do it thorough instead of adding a lot of different content but less thorough and relevant. At the same time we needed to consider the complexity of implementing the different question types into the game. We understood that we did not have time to implement all of them, so we decided to look more into which question types could be used to answer questions from the different topics:

Topics in system development	Multiple choice	Click on image	Draw on image	SQL-question	Number compare	Text parsing	Drag and drop
Top-down or bottom-up approach to system development	X	X					X
System development strategies	X	X	X		X	X	X
Development platforms and software	X	X				X	X
Relational database	X	X	X	X			X
Data models	X	X	X				X
Normalization and redundancy	X					X	X
Constraints	X		X			X	X
Data representation	X					X	X
Designing user interfaces	X						X
Evaluation information systems	X					X	
Knowledge about different issues involved when making contract between client and the developer	X					X	
Price and estimation	X				X	X	
Law and ethics	X					X	

Table 2: Question types vs. educational content

This table is generated based on our opinions on how well suited the different question types is to make questions about the different topics. The X's indicates a good match on question type and topic.

What question types to implement in our game

As we see from fig x, the multiple choice question type can be used to ask question from all the topics we wanted to have in our game. When considering how easy it is to implement such a question type, we quickly decided that we were going to implement Multiple-choice questions. We also decided that the question type would be a lot better when supplemented by images, so we also chose to implement this.

Further we see that Text parsing was a question type that could potentially be good. The draw back with text parsing is that it is quite hard to implement in the game. For text parsing to be good one would need very good and complex regex programming. This could take some time to program in order to get the question type working in an acceptable way. Therefore we chose to see if there were other question types we would rather implement.

Along with text parsing, drag and drop is potentially good but very hard and time consuming to implement in a good way. The question type in itself would probably prove to be great as it stimulates analysis and problem solving skills. Considering the complexity of implementing we chose not to use drag and drop.

Another question type that stimulates analysis and problem solving skills and individual thinking is Draw on image. As we see from table 2, the topics this question type can be used in is rather narrow, but we believe that the topics it covers is rather central in the curriculum so we decided on trying to implement it.

We found out that if we were going to implement Draw on image, it would also be quite easy to implement click on image. The programming code for those two question types are rather similar, so we decided to also implement Click on image.

When we considered the number compare question type we decided that its use were rather narrow and could always be replaced with a variation of multiple choice question. We chose not to implement the number compare question type.

As we can see the SQL question type is basically only useful for testing SQL knowledge. When we made our considerations on what question types to implement, SQL queries were a central topic in the old curriculum. Therefore we chose to try to implement the SQL question

type even if it could prove to be rather difficult.

Top-down or bottom-up approach to system development

To make questions that teach students the difference of top-down versus bottom-up approach can prove to be quite difficult. It is quite easy to give a question-text and answer alternatives, but this only check if the students already know the difference. If one should make good questions on this topic it should include a good way of giving proper feedback with good explanations. As the curriculum of INF1050 changed, the focus on top-down vs. bottom up approach lessened. We therefore chose to not put any focus on this topic in our questions.

System development strategies

This topic is rather central in the process of system development. We therefore chose to add a lot of different questions that makes the students identify the different forms of system development methods and strategies, and further test their ability to identify the benefits and downsides of the different development methods. On the other hand we chose not to test the students if they could pick out a development strategy and explain why the chosen strategy is suitable for a given project text. The reason for this is that there are several different strategies that are suitable for the same project, and for us to program a game that gives feedback that depends on a lot of different conditions would be very time consuming and difficult.

Development platforms and software

From our own experience and earlier exams, development platforms and software is something that is not explicit tested in the exams of INF1050. («INF1050 exams», 2007) This subject is more like an introduction to the students so they understand the basics on how a computer works. As this is relatively easy knowledge to test the students in, we chose to add a couple of questions from this topic in the game.

Relational database

When we first started making questions for the game relational databases and SQL were a central topic. We therefore developed a SQL-checker that parsed SQL input and checked it up against the correct answers. Before the curriculum change in INF1050 the students were required to make web sites that interacted with a SQL database. As the curriculum changed this was left out and it is still an uncertainty if SQL should be a part of the revised curriculum. As a result of this we chose not to include any SQL in our questions. We added some easy

questions about basic relational databases as this still is a topic in the course.

Data models

Since the creation of data models often require the students to write in objects manually and that it is quite hard for us with our current question types to check manually written text, we chose to put very little focus on this topic in our questions.

Normalization and redundancy

As this topic is pretty basic and most of the students are required to have knowledge about this topic on their exams, we chose to add a couple of questions similar to an earlier exam given in INF1050. (ibid) Testing the students in questions that are similar to those they get on their exams is also a good way to prepare the students on how their exam situation will be, and through this aid them in doing better.

Constraints

The drawing questions type in our game is a good way of testing if the students understand how to use constraints. We therefore added some questions that make the player draw in lacking constraints on an image of a database. There should be no problem in adding more questions that test the player in the use of constraints, but in light of our limited time on making questions we chose not to.

Data representation

The book covers how various data representations can be used to reflect real world concepts and explain how they are used in an information system. The students are expected to be able too choose appropriate data representations for information presented in a project. They should be able to discuss the benefits of using one form of data representation rather than another in a given example. The book also covers identifiers and various levels of information carried in data representations. An example on this is how our social security number also contains information about gender.

Designing user interfaces

Our current three question types is not very well suited for testing or teaching the students in design of user interfaces. As use-case diagrams are linked to the design of user interfaces we made some questions on this instead. For further making more questions on this topic our

click on image question type could prove rather useful.

Evaluating information systems

To teach and test the students on this topic in a good pedagogical way, the students would most probably have to write in text manually. As we haven't made any text parser we chose not to add any questions about this topic.

Knowledge about different issues involved when making a contract between the client and the developer

When it comes to knowledge about contracts there are many considerations one need to take into account. This makes it very hard to make questions that have one set right answer, and therefore our question types are not particularly well suited to test this topic. As a consequence of this we chose not to put any focus on this topic in our questions.

Price and time estimation

Earlier when we took the INF1050 course this was a topic the professors meant that this was a topic system developers could gain a lot from by improving in. Therefore we chose to add some questions related to this. As price is very varying from project to project, we rather chose to put focus on time estimation. We added some questions that require the students to sit down with pen and paper and draw diagrams in order to answer the question correctly. These questions are also very similar to the ones the students get at their exams.

Law and ethics

As this topic is rather easy to make questions about we have included this in our game.

4.3.1 Curriculum changes in INF1050

The curriculum of INF1050 changed while we were writing this thesis. It went from a practical course based on a large group assignment which required some programming, PHP and SQL skills, to a more theoretical course. As we were dependent on using the INF1050 students to test our game, we chose to adapt our content to reflect the changes made in the curriculum. The biggest change was that they were uncertain if they were going to have SQL as a topic in the new curriculum. We therefore chose to remove the SQL questions from the game. Before we conducted the tests we compared the new curriculum with how far the course had progressed and removed questions that the players had little chance of answering correctly.

In its current form the students are encouraged to discuss aspects of system development to identify benefits and drawbacks related to certain system development approaches. This made it more difficult for us to choose what to include in our game, because it is so hard to design something that automatically correct such assignments. In our game it is rather easy to test trivia content through multiple choice questions but the potential learning from such questions are rather limited compared to questions were one have to analyze a picture and correct it. Mainly we focused on adding system development knowledge that is easy to test through our three ways of giving questions.

4.4 Game characteristics

Based on the educational content we decided to implement, we will now present the design characteristics and functional requirements we feel are necessary to include in our game. Our focus here will be on characteristics and requirements that will enable us to fuse the educational content with the motivational aspects. These are based upon the knowledge we learned from the thesis' we examined in chapter two and through our own experience with system development, user interfaces and games.

4.4.1 System design characteristics

When developing a software system there are several important subjects to consider. In this chapter we have outlined 4 main topics we think are the most important for our game development. We have mainly focused on topics that we believe important for our educational game to succeed.

User friendly

It is very important to our system that it is easy to use for the player, assignment creators and future developers. As the goal of this system is to teach educational knowledge, we need to make sure that the players do not quit the game because it is too hard to understand how to get hold of it, how to install it or how to play it. To ensure this we need to make an easy to understand Internet page that explain how to get started with our game in a step by step tutorial. Further we need to minimize the technical issues that can complicate the installation of our game. When it comes to the actual game, we need to make the user interface as self explaining as possible. The places, menus and windows in the game need names that reflect

what the player can do in the different situations that occur in the game.

Extendable

We will write this game in such a way that it should be rather easy to adapt the content to cover other things than just system development. As this game is a part of a master thesis, other people will be interested in our findings and might want to further develop our game. Therefore it is important that structure the code good and that we document our code in a good way, so later developers do not waste time on having to understand our code and work before continuing on the development.

Reusable

Even though this game is written and tested on people learning system development, our goal is that the game can be used to teach different types of knowledge. Because of this, it is important that we write the code in such a way that it is easy to change the knowledge theme of the game. Again it is very important that we document how to do this in a good way. The easiest way to ensure this is to make it easy to edit the story, questions and names on the places in the game.

Entertaining

To get the players to play our game and learn something has to be the key element of this system. To incorporate this, it is important that the educational aspects of the game do not interfere with the entertainment. What we hope to do is implement the educational aspects in a subtle way, so they do not seem cumbersome. It will be interesting to implement the educational aspects of the game, as we try our best not to lessen the fun factor of the game. I deem this will be our greatest challenge in this design process. This is why our preliminary plan, is to incorporate the fun elements of game first. Only when we feel we have created a game that is fun and challenging will we implement one educational aspect at a time.

4.5 Functional requirements

4.5.1 The game

Type of game

The game we will develop is a single-player educational game where the player controls the life of a fictional avatar. The player needs to guide the avatar through life, giving the avatar education, work experience and earning money through working. In order to advance in the game the player will be presented with educational questions that needs to be answered correctly in order to advance in the game.

User interface

The user interface of the game should be a graphical window user interface with almost self explaining information boxes and menus. It is important that the users of the game do not get demotivated by an advanced user interface. It is also important that it is well documented in a help text in case the user is unsure of the different functionalities featured in the user interface. The game's user interface should also support both Windows and Linux as these are the most common operating systems used by the game's target group.

User input

The player should be able to answer the games question through the use of the computer mouse. In addition the mouse and keyboard is used throughout the whole game to interact with the user interface.

Answer educational questions

The game will be able to give the player educational questions based upon how far into the game the player has advanced. As feedback is an important aspect of learning the player should also receive appropriate feedback based upon the answers given.

Scoring and goals

To give the player a competitive edge to the game-play it is important that we implement some way of giving the player feedback on how good he/she played the game. We will try to implement 4 different goals (namely happiness, money, work-experience and education) that

the player needs to reach in order to complete the game. At all times during the game-play the player will be able to view how far he/she has advanced to reach the given goals. We also need to give the player a score at the end of the game to give feedback on how well the player did. This will also encourage the players to compare scores, compete between each other and through this play the game more and learn more knowledge.

4.5.2 The assignment creator

The purpose

The purpose of the assignment creator is to keep the game evolving at all times. With this feature the persons that have access to the assignment creator can totally change the question content of the game and also be able to adapt the content of the game to teach different types of knowledge.

Availability

The assignment creator should not be available to the regular user of the game, but rather be an administrator feature. This will ensure that the students have to learn the question answers through reading literature rather than checking the questions solutions through the assignment creator.

User interface

An author of new questions should be able to add questions through a well known windows-like environment. The author should also be able to choose where in the game he/she would like to add the questions.

Administrating the questions

The assignment creator should let users with administrator rights add, view and change questions in the game. Further it should allow administrators to chose where in the game the different questions will be given, and change the locations of the questions. Another feature that will be added is the possibility to make an exam. An exam is when the player has to answer several questions following each other correctly in order to advance.

Add images

The assignment creator should be able to add images to the questions. It should also be able to

give points of interest to the images, so it is easy to make the different kinds of question types without manually having to enter image coordinates etc.

4.6 Summary

We have defined our educational content to be based upon the curriculum of INF1050. We gave an overview of the INF1050 curriculum as we researched it in late 2007, and how it changed at the start of 2008. INF1050 covers the theoretical basics of system development, and includes topics as: Data representation, Constraints Normalization and redundancy, Data models, Relational database, Development platforms and System development strategies. Based on this knowledge we explored different question types we could implement in our game and landed on: multiple choice, image identification, drawing and SQL.

Based on these choices we decided on the system specifications for our game development. These specifications are founded on the idea of merging the educational content with the motivational aspects from chapter two. The system design characteristics we defined are: user friendly, extendable, reusable and entertaining. Further we set up the functional requirements for the game and the assignment creator that administrate the questions that the player is given in the actual game-play.

Chapter 5
Technologies

In this chapter we will give a brief overview over the different technologies and programs we used and some we considered using, to develop our educational game.

We start out by telling more about what considerations we took into account when deciding upon what programming language to write in. Further we go more into the different web technologies we used to collaborate on shared texts, and how we deployed our game over the Internet. We continue with giving an overview over the programs we used for the actual game development, and sum up with telling more about technologies we didn't use and why.

5.1 Programming language

The first thing we did when we knew we were going to write our educational game from scratch, was to was to decide on which programming language we where going to write in. As we knew we had limited time to finish our master thesis, we decided that we needed to write in a language that we both had prior knowledge to. This stripped our alternatives down to four different choices: C, Java, Perl or Python.

As one of our goals with the game was to implement system development assignments, we needed a language that supports the use of drawing tools. Another feature we wanted to be able to implement was to show an image to the player and let the player be able to click on the image and get a response accordingly to where he/she clicked. These features are needed to make the game as realistic as possible for the user. A purely text based interaction with questions and riddles would not suffice with our intention for the game. From our earlier knowledge we knew that Java have a large library of classes that support these features, and that they are relatively easy to use.

We also liked the idea that the users should be able to write in text manually and that the game should be able to parse it and pick out important elements of the text. A good way of parsing text is to use *Regular Expressions* (REGEX).(«Regex», 2004) Regex provide an easy way of recognizing and finding specific character- or word-phrases in texts. Perl and Python have excellent REGEX functionality and are very good languages for parsing text. Lately Java and C have improved their libraries and added this functionality as well.

Another important issue was that our game needed to be easy to install and use. Since our potential user group is students, and the game we develop primarily is intended as extra material to the regular lectures and course assignments we needed the game to be easily available for the users. Ideally it should be able to play the game over the Internet without having to download or install a lot of different programs.

We looked into several different solutions to achieve this, and found out that using Java with the Java Web Start technology would probably be the best solution.

5.2 Web technologies

5.2.1 Java applet

Java applets («Java applet», n.d.) are developed by Sun Microsystems («Sun», 2008) and were introduced with the first Java language in 1995. Java applet is Java code that can be integrated in HTML code. When the HTML code then is viewed with a Java-technology enabled web browser, the applet code is transferred to the local machine's Java Virtual Machine (JVM) and executed in a sandbox there.

The main advantage by using Java applet in our project would be that the users do not need to download and install the game itself but rather play it directly in the web browser. The disadvantages would be that the user would need constant Internet connection in order to play our game. Another major disadvantage is that our target group for the game often is located at universities with computers where they do not have administrator rights. This could potentially exclude people from playing our game and thus limit our user group. Therefore we chose not to use Java applet to develop out educational system development game.

5.2.2 Java Web Start

Java Web Start («Java Web Start», n.d.) is a framework developed by Sun Microsystems that allow Java applications to be run through a web browser. Java Web Start has been included as default in the Java Run-time Environment since J2SE, and thus most people that have Java installed on their computer got access to Java Web Start.

In difference to Java applets, Java Web Start does not run the application directly in the web browser but rather in a sandbox on the client computer. The end-user download a JNLP file that includes information about where the jar files for the Java program is located. Then the jar file is downloaded and executed accordingly to the information in the JNLP file. One of the main benefits of using Java Web Start is that it is easy to write a Java program for a stand alone computer, and make it available over the Internet without having to rewrite large chunks of the code.

Another huge advantage with using Java Web Start is that it overcomes much of the difficulty with problems that often occur with browser's Java plug-ins and Java Virtual Machine versions when publishing programs over the Internet. Java Web Start supports Internet Explorer 4 or higher and Mozilla, but any browser that can launch JNLP files can run Java Web Start if MIME-type association is set correctly. We chose not to use Java Web Start because we tested it on several computers campus at the University of Oslo, and found out that since the regular student lack computer administrator rights, Java Web Start fail to initialize correctly.

5.2.3 Google Docs

We used *Google Docs* («GoogleDocs», 2008) for most of our document writing, taking notes, and discussing further progress. When multiple people work on the same project it is essential to have an asynchronous form of communication, and since we were already using Google Docs for our document collaboration, it was a natural choice for simple asynchronous communication as well.

Google Docs is a free online collaborating writing tool that let the users share documents, spreadsheets and presentations over the Internet. The main collaborating artefact is the feature that lets several people located at different geographical locations edit the same document simultaneously.

To get access to Google Docs one need to register a Google-account (often referred to as a Gmail account). Then one is given a personal hard disk space on their server and is free to upload documents or create new ones from scratch. Every document is private until one chooses to either publish them so other people can view them, or invite other collaborators with Google accounts to the document and thus allow them to edit it.

Google Docs comes with an easy browser supported editor with the same functionality as regular text editors such as Word and OpenOfficeWriter. The functionalities include spell-check, regular text formatting etc. One can also upload images that one can associate and add to the document.

When editing a document one is shown a list of user names over the other users that are working on the same document at the same time. If a conflict should occur, for instance that two users edit the same paragraph simultaneously, a pop up message is shown to one of the users informing him/her about the conflict and neglects the changes made.

The Google Docs database is constantly backed up, and because of this it is a relatively safe way of preventing information being lost due to hardware failure etc. Another nice feature with Google Docs is that one can download the finished documents to your local computer in different file formats such as: Word, OpenOffice, RTF, PDF, HTML or ZIP.

5.2.4 HTML

Hyper Text Markup Language (HTML) is the most used publishing language on the Internet. HTML is standardized by the *World Wide Web Consortium* (W3C) («W3C», 2008) which is the main international standards organization for the World Wide Web. In our project we used it to develop Internet pages to aid the users download our program and read the tutorial.

5.2.5 PHP

PHP: Hypertext Preprocessor (PHP) is a HTML-embedded scripting language that allows web developers to write dynamically generated pages. PHP is developed by *The PHP Group* («PHP», 2008) and is not standardized by W3C or any other such Consortium. As of today The PHP Group serves as a de-facto standard for PHP. We used it in our project to develop an online Internet web survey that sent the results straight to our e-mail.

5.3 Development tools

5.3.1 Eclipse

Eclipse («Eclipse», 2008) is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and run-times for building, deploying and managing software across the lifecycle.

Eclipse SDK includes Eclipse Java Development Tools, offering and Integrated Development Environment (IDE) for Java with a built-in Java compiler and the full model of the Java source files. This allows Eclipse to provide features like advanced code re-factoring, debugging and built in syntax checker in the code editor.

We used Eclipse basically as a program to organize our Java project in a well arranged way. Among several things we used in the program, Eclipse includes a package explorer, code-editor, built in compiler and debugger. Eclipse also includes nice graphical overview windows that easily let the programmer find methods and classes in a quick way instead of having to scroll through the whole file.

5.3.2 Subversion

Subversion («SVN», 2006) is an open source revision control system. It is used to keep track of changes in files and directories, and allows for multiple users to make changes to the same file or file structure. Subversion stores files in their current form, and older revision, in a repository. When users are satisfied with a change made to a file they commit those changes in the repository, and likewise they can check out files to get updated changes from other users. The most important functions of revision control systems, is their ability to restore older versions of files, and resolve file conflicts. Restoring older versions can be important. Ff changes that have been made to a file later appears to be wrong and perhaps break the program or application they belong to, restoring an older version of the file might correct the problem. Conflict solving is essential for allowing multiple people to edit the same file. Sometimes changes are made that do not reconcile, for instance two users may edit the same word in a text file, to a synonym of that word. Subversion will then let the users know that there is a file conflict, and suggest solutions.

We used subversion during our entire development process to handle our program files. It allowed us to work on the same files in the program, even from different locations. Whenever

we wanted to work on the game, we simply checked out the latest version of the software from the repository, and check the changes into the repository when we were done. Subversion also worked as a continuous backup of our system as we always had an up to date version of the game stored in the repository.

5.4 Using existing software

During our design process it was important to look into the possibility of using existing software to facilitate our tests. To find out if we could use existing software we looked into the general benefits and drawbacks of using such software. Some of the benefits are quite obvious. By saving a lot of development time on creating the basics of our program, we could have more time to focus on implementing the educational content we want. Some of the game generators that are available, like Game Editor («GameEditor», 2008), provides the user with methods of easily creating graphics and animation that help create visibly pleasing games. By using existing editors we could create a game that would, at least in appearance, be a lot more advanced that what we can create from scratch.

There are some huge downsides to using existing platforms however. The time used to familiarize oneself with the software might easily exceed the time saved on implementation. The limits to the software chosen will also to a bigger extent limit the project. Adding or changing functionality of existing software might be impossible (legal limitations), or at the very least extremely time consuming (editing unfamiliar code is more time consuming that editing one's own).

As we started our project we were made aware of a student that was making a game generator specifically designed to create system development games. In her thesis «A game generator - the framework for an educational system development game» Anne Chr. S. Landro (Landro, 2007) created an input driven game generator with support adding modules to test advanced user feedback. Her thesis focuses on creating a generator with the capability creating games that simulate the system development process, and less on the motivational benefits of computer games. As her project was not finished when we started on our thesis, and we were uncertain of the outcome of her work at the time, we did not feel we could base our project on her game generator.

There are some educational games that have been made for students at a university level, or

that can be applied at this level of education. An example of such a game is Colobot («Colobot», 2006). A game that despite its childish appearance is capable of teaching advanced programming techniques. The game consists of using a language similar to that of Java or C/C++ to program robots to colonize planets. The robots can perform tasks such as resource gathering and elimination of hostile alien life. Since the robots have very limited AI, the players have to program every menial task the robots perform. This ranges from simply moving 20 feet forward with a «move(20)» command to programming advance search and destroy algorithms. Having tested the demo ourselves, we could definitely see the potential for this game used as a supplement to the early programming courses at our university.

However, since our thesis revolves around testing the motivational benefits of using educational computer games at a system development course, we did not find any existing software that was a close enough match to what we wanted. With the importance for us of having the ability to implement the motivational aspects we had research we decided that implementing our software from scratch would best suit our needs.

5.5 Summary

Deciding upon what programming language to develop in is very crucial for the project. We chose to develop our game in Java, because we have good prior knowledge to it and believe that it is advanced enough to cover our needs.

The important web-technologies we used in the development of our game and this thesis includes Google docs, HTML and PHP. We used Google docs to collaborate on writing the actual paper. HTML and PHP were used to make the download instructions for the game, the tutorial for the game and the web survey.

As this project is quite large and includes several thousand lines of Java code, we needed some way of keeping control over it. We used Eclipse to aid us in writing the code and keeping it organized and structured. As we were two persons programming in the same Java files we also needed some way of making sure that the code we edited was the latest version. Therefore we set up a Subversion repository to keep control of the changes we made during our programming.

After reviewing what existing software was available to us, and how it fit our project, we decided upon designing the software from scratch. This was mainly done to ensure that we

could implement the motivational aspects we had identified in our research.

Chapter 6
Design

In this chapter we will discuss the design elements of our game. We will explain the different elements in the game and how they relate to our overall goals of educating and motivating with the game. The focus of this chapter is on the different aspects of the design and how they directly affect the player and the gameplay.

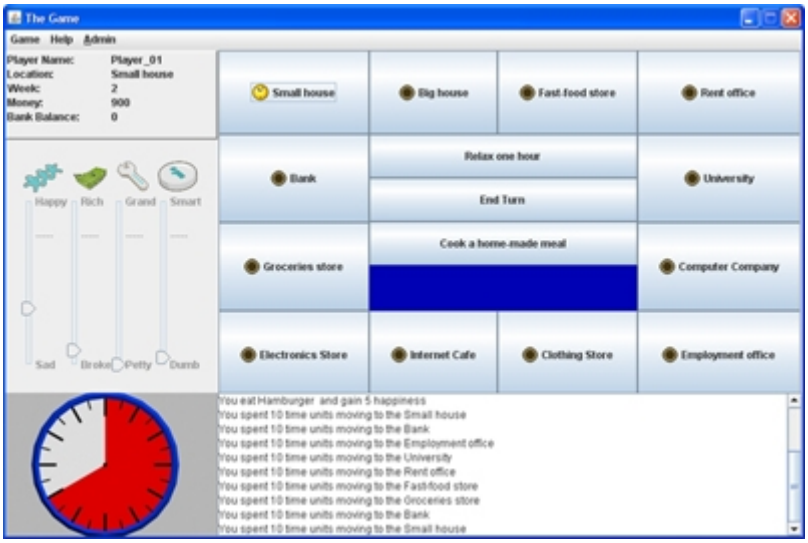


Illustration 8: The game's user interface

6.1 The elements of the game

Money

The game uses a fictive currency for money that is not directly connected to any real world currency even though the symbol for American dollars is used in reference to it throughout the game. Money is one of the four main statistics in the game. Almost every action the player can perform in the game is affected by money in some way. As this is a career based game it was only natural to make money an integral part of the game. Money is gained through

working and used to perform most other actions. It can either be carried around on the avatar or stored in the bank for rent interest and security against mugging. Since using the bank takes time, the player has to decide when it is most wise to bank money, or when it is needed to pay bills. Money management is performed by the player by choosing how much time in any given turn should be used to work. This choice needs to be made based on what expenses are expected, and how much time is needed for other activities.

Happiness

The avatar is given a value based statistic for happiness. It starts out on a mediocre level, reflecting the state of the characters life upon starting a new game. This value is affected by how well the player manages events in game, and is also an important statistic when calculating the end score. Managing happiness is less decided by random and scheduled events than money and used more like a control value. Players are punished with negative happiness when breaking game rules and awarded by abiding by them. Key issues that effect happiness includes eating, resting and work overload. The player needs manage happiness by setting of time every turn to eat and managing work versus relaxing; if they forget to eat they will get a happiness deduction. This also makes happiness an integral part of money management, as one can sacrifice some happiness to work more, or work less to get more leisure time.

Work Experience

Work experience is a value that is meant to simulate real world work experience. It is normal when applying for work to show a resume of previous work experience, and it is usually a deciding factor in job applications in general. Work experience in our game is simplified. It is a simple numerical value that represents the experience you have from your best work position. Every work position has three values that affect work experience, the work experience needed to get the job, the work experience obtained from working there and how long you will have to work there to achieve the experience. These simple values let us create several parallel career routes you can climb in order to reach the top. Some jobs can give a high work experience reward but low income, and others can reward the player with a high salary for its requirements, but the work experience takes a longer time to achieve. Work experience is the key feature that ensures progress in the game. Once gained, the player can never loose work experience, and every time it increases it opens up for new job opportunities that provide the player with new challenges in order to complete the other requirements of the

available work positions.

Education Level

In addition to work experience, most jobs need a set amount of education level in the three different venues of education in the game in order for the player to get the job. Education is a progressive stat similar to work experience, but it differs in that it is not something that comes naturally with normal turn actions. The players have to make an active choice to seek out the University in-game when they feel that they can spare enough time and money for the course. This further complicates the choices the player has to take in order to balance money and happiness.

Time

Time is the final statistic that the player has to manage. The game is divided into turns that represent a week. During this week the player has a set amount of time points to invest in different activities. By effective management of the allotted time each turn, the player can better manage the other statistics. There are no random elements or scheduled events that effects time, so the players are in full control of how they want to manage it. Certain actions within the game can deduct time from the player without giving any reward. This includes applying for work positions that you can not get and applying for assignments you do not answer correctly. Punishing wrong actions in the form of time deduction effectively deduct from all the other stats as the player has less time for work, education and rest. Forgetting to eat during a turn will punish the player with less time to spend the next turn as well as the happiness deduction.

Furniture, Housing and Clothing

In addition to managing all the statistics described above the player has to choose then to invest in one of the many commodities available in the game. Home cooked meals become available upon purchase of a fridge and groceries from the local groceries store. Until the players can afford such an investment, they are forces to eat fast food, which affords the player less happiness and is more time consuming. Upon purchasing a fridge the player may want to upgrade to better housing, as it comes with better security, which in turn results in less chance of getting robbed. If a player is robbed all the furniture will be removed from the player.

Random events and complexity

All games need a level of uncertainty in order to be called games (Kramer, 2000). Uncertainty is what makes games enjoyable to play more than once, since the player can not be certain of the outcome before playing. The uncertainty aspect in our game comes from complexity and random events within the game. With the number of choices the player can make during the course of the game, it is highly unlikely that a player could play the game in the exactly same way more than once. This coupled with some random events, ensures that the gameplay is different each time. The random events come in the form of actions that can happen in the game that are beyond user control. There is a chance to get mugged or robbed while walking around in the game and after every turn there is a random weekend event that affects the player's stats. In addition there are some random events that the player has some limited control over. For instance, if the player works too much, they might get sick and get a medical bill. The player has to make a choice to either risk getting the bill or playing it safe and spend less time working.

6.1.1 Avatar

The avatar represents the players in the game. The player can choose the name for their avatar upon starting the game. This name is used for all the feedback to the player. This helps to personalize the interaction with the player. The visual representation of the avatar in the game is that of a small smiley. It changes appearance as the player gets more successful in the game and can afford more expensive clothing. It starts out as a befuddled looking face and evolves to a confident smiley complete with sun glasses. As the players moves around in the game they can see their position indicated by the small smiley moving around the game world.

The avatar is given a small background story to help create a feel of complexity to the game world. The story is kept vague as to make it easier for the player to relate to it. It tells of an aspiring young person with dreams of becoming a partner at the local computer company, and lays a backdrop of how you have just moved to a new town and find yourself pressed for cash with limited education and work experience. The story presented also hints at possible actions the player may start out with to get into the game

6.1.2 Assignments

Whenever players take a course at the SchoolPlace, called University in the game, they are presented with an assignment. These assignments need to be answered correctly in order for

the player to increase in education level. There are currently four different types of assignments in the game, and a system is set up to allow for easy implementation of additional assignment types if needed. We will briefly go through how the assignments are presented to the players and what is expected of them.

Multiple Choice

These assignments consist of a question, an optional image related to the question, and a set of answers ranging from two too six per question. Based on the information given in the question and optionally presented in the image, the player has to choose one of the answers listed before submitting the answer by clicking a button labeled "OK". This assignment type can be used for asking any theoretical question with a defined answer, and is the most used question type in the game in its current implementation.

Illustration 9: Example of multiple choice question

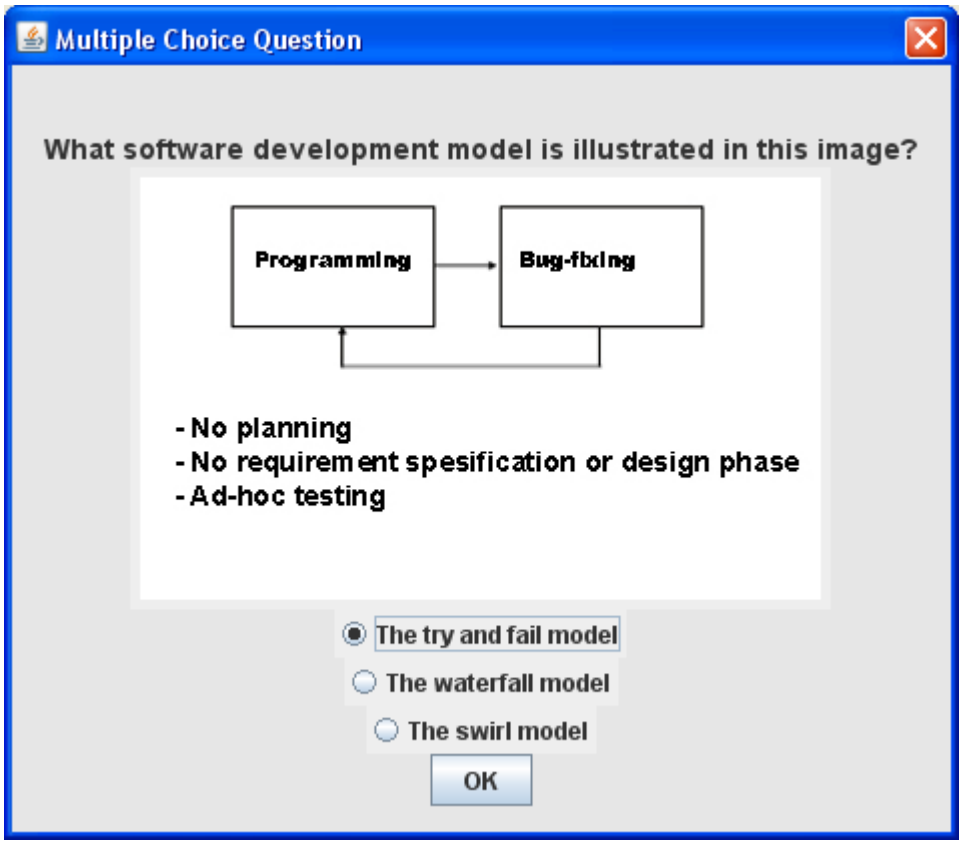


Illustration 10: Example of multiple choice question with image

Click On Image

The Click On Image assignments consist of a question and a corresponding image shown on the screen. Based on the information given in the question, the player has to choose the correct coordinates on the image to click on. The player is only given one chance to click on the right coordinates of the image, and once clicked the image will disappear and the player will be given feedback on whether or not they clicked on the right location. These assignments are used to test the player in diagram and image recognition. It is common to present students with diagrams to give an easier understanding of how different theoretical elements are related to each other, and the Click On Image assignments can be used to ensure that the students have understood these diagrams.

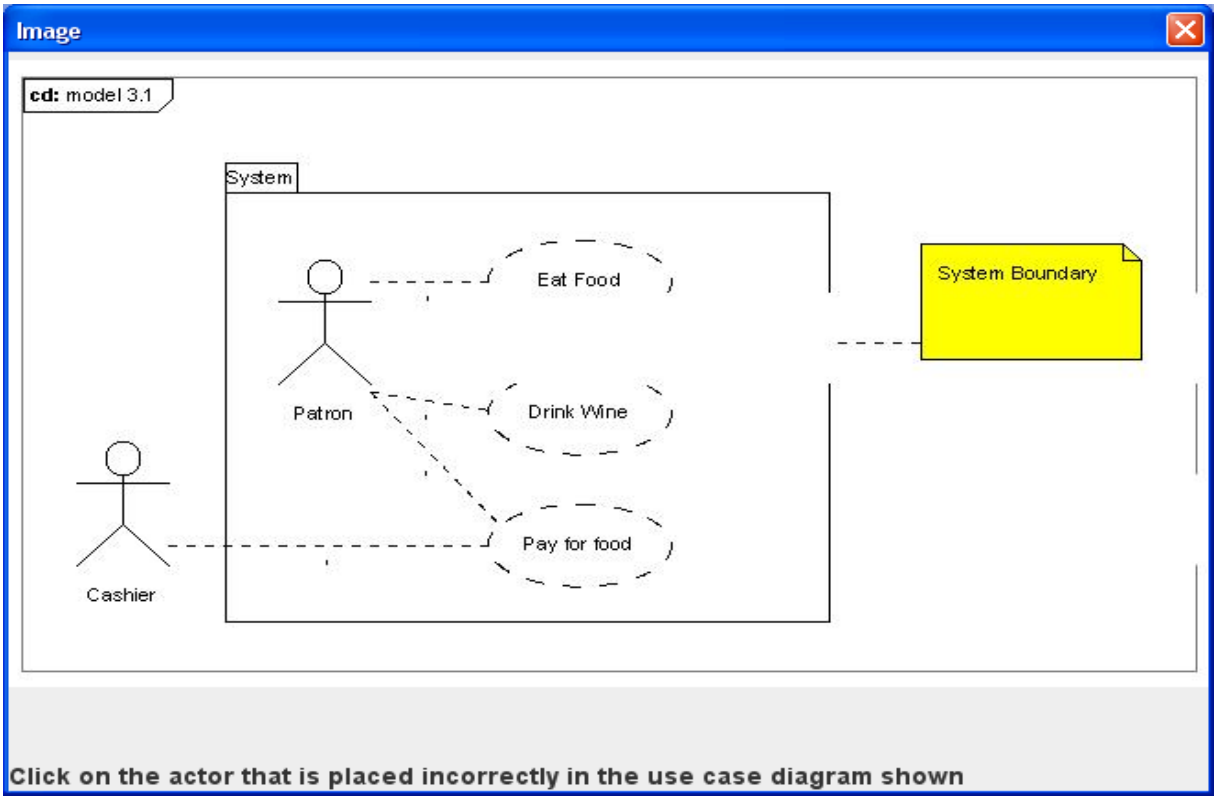


Illustration 11: Example of click on image question

Draw On Image

Draw On Image is a more advanced version of Click On Image. The players again given a question and a corresponding image, but are asked to draw the answer on the image. Given the nature of the image, the players are given the opportunity to draw on the image as many times as they like until they are satisfied before submitting the answer. As the player draws on the image they create a thick red line over the image which is easily visible. This question type is used to test for more advanced diagram understanding. The player can be asked to draw the missing parts of a diagram, connect objects with a certain similarity, or draw in certain elements like constraints on a database diagram.

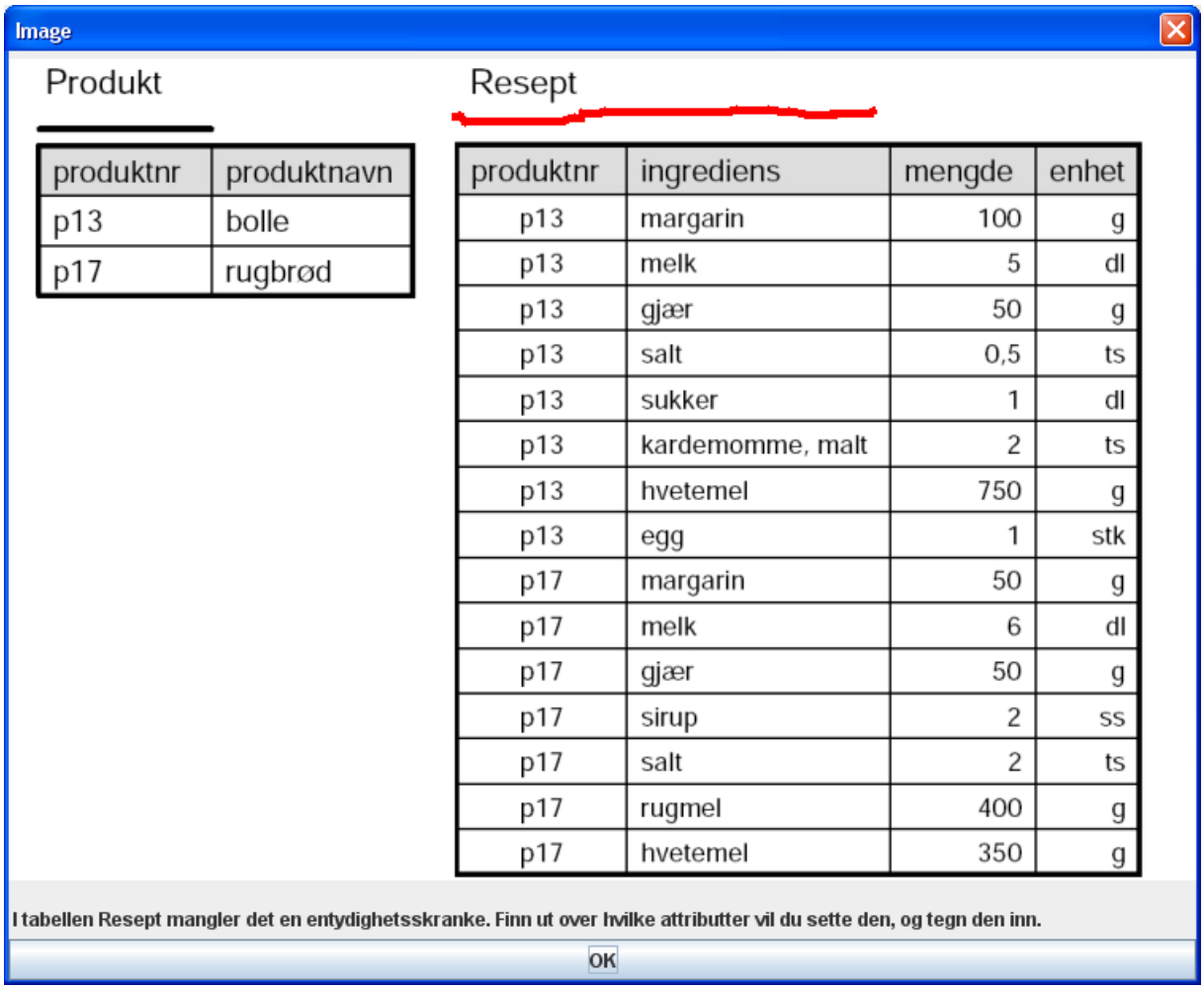


Illustration 12: Example of draw on image question

SQL Question

SQL Questions were designed to be the hardest assignments in the game, but were later removed from the current implementation because of the changes in the INF1050 curriculum. SQL Questions present the players with a set of database tables and a question with the task of getting some specific information out of the tables using normal SQL statements such as JOIN and WHERE. The answer given by the player would then be parsed by an SQL parser, and only a correct SQL statement would be accepted.

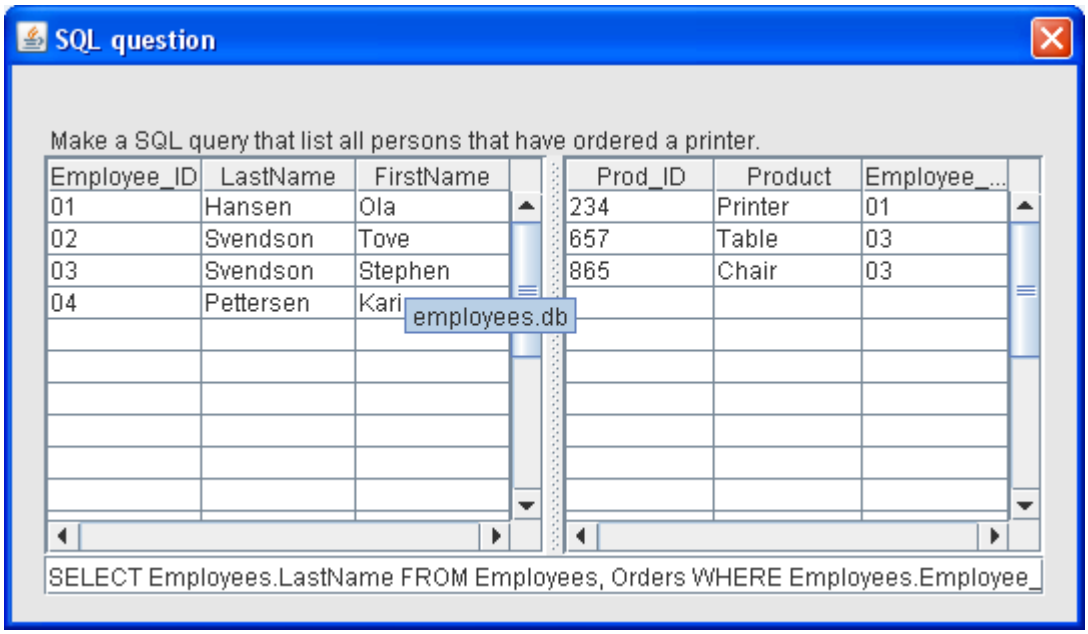


Illustration 13: Example of SQL question

6.2 The flow of the game

When a new player starts out in the game, he or she is given a limited amount of funds and apartment with one months of prepaid rent. The player is presented with a background story which sets the ultimate goal of becoming a partner at the local computer company. In order to reach that goal the player has to climb the career ladder in the game in order to get the required work experience needed for this job. The player has regular expenses related to food, clothing and rent. In order to maintain these expenses one will have to climb the career ladder sufficiently before the initial money buffer runs out. As the player is progressing through the first few jobs, it quickly becomes apparent that one needs education to get better jobs. The player then has to go to the university and apply to a course. Each course has an array of mini-game like assignments based on the curriculum from INF1050 with increasing levels of difficulty. The player needs to answer these correctly in order to increase the education level for the chosen course. If the player lacks the knowledge to pass the test, he/she will have to read up on the course literature in order to find the right answers. We believe that this way of implementing the learning will encourage the player to learn system development. The player is given a short term goal that he needs to reach, and this will motivate the player in acquiring knowledge.

In the following picture we show the main elements of the user interface.

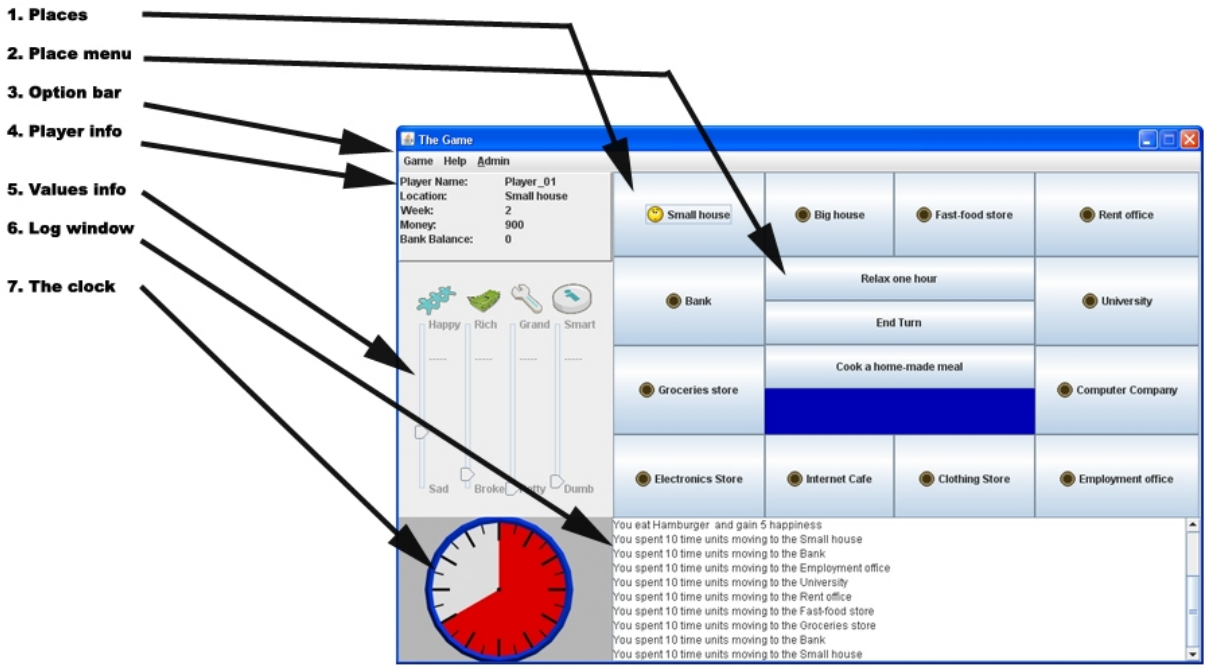


Illustration 14: The main game window

- **1. Places -** These 12 buttons represent the different places you can visit in the game. The smiley is representing your avatar and shows where you are located.
- **2. Place menu** - The centre button row represent actions you can perform at the location your avatar is currently visiting.
- **3. Option bar -** Standard drop down menu where you can open different information windows, start a new game and such.
- **4. Player status -** Information about your avatar and time.
- **5. Values info -** Show the 4 different goals in the game, and a bar that indicates how many points you have in the different goals. When you gain enough points to beat the game in a certain goal the icon will change to a glowing icon. The amount of points needed in the different goals is indicated by the dotted line.
- **6. Log window -** This window will log all actions in the game.
- **7. The clock -** Show how much time that is remaining of the week. When it is fully red, the week is over, and you are moved home to your home place.

6.3 The gameplay

As the game progresses the player will acquire better work titles and higher salaries, but the expenses also increase. Better paid jobs require better clothing, rent expenses go up as one upgrades housing and food expenses increase as one is in need of better food to maintain a high level of happiness.

To succeed within the game the player is required to cleverly manage his time. Each turn the player is given a certain amount of time units. If the players have done something wrong in the previous turn, like forgetting to eat, they will be penalized with less time units for the successive turn. Each turn has to be thoroughly planned to minimize the amount of movement that need to be done. Moving from one location to another requires time units, and having to visit the same place more than once during a turn is a waste. With that in mind, the player has to choose how much time to spend on the different actions in the game. If the player is in need of money, if for instance the rent is due soon, the player has to make a conscious choice to spend more time working for a few rounds. This however means that the player has to neglect some of the other important values in the game, like happiness and education, and they will fall behind on those. After the bill has been paid, the player has to correct that deficit, which means choosing to spend more time on education, leisure and spending money on more expensive food. This constant effect each value has on other values, in addition to random events and timed events in the game that affect them, means that every turn in the game is different. It is by truly mastering this balance that the player can excel within the game.

6.4 Motivation and entertainment in our design

Entertainment value

The entertainment value of the game is strongly connected to the challenge of managing the player's statistics. The challenge is founded on a progressive career based game. This in turn incorporates a lot of the motivational aspects that Gee identifies in his paper (Gee, 2005). It gives the player a sense of identity with the game through the avatar that they name after themselves or with a name of their choosing. Giving the player the power to perform actions that change how the world and the avatar look within the game fulfils Gee's criteria of manipulation. This is done in our game through buying new homes, changing clothes and applying for new jobs etc.

Success within the game is directly connected to the player's ability to plan and perform

strategies. This involves system thinking which is one of the criteria for Empowered Learners.

Informational flow and uncertainty

The informational flow of the game is also according to the teachings of Gee. Withholding information about elements such as the exact criteria for obtaining work positions gives players a sense of uncertainty and curiosity. "Am I able to get that job at the bank now?" When in need of information to make choices within the game, it is provided "on demand" and "just in time". The "on demand" portion is fulfilled with the statistics and information window within the game. If the player is uncertain of when the next rent is due, he can open this window and get this information. All other feedback is given "just in time" whenever players perform or try to perform an action, they are given feedback on the results or why the action could not be performed.

Goal, score keeping and competition

In order to motivate the players to not only play, which comes from the entertainment value of the game, but also try to excel while playing it is vital to have goals which the player can try to achieve. There are several goals within our game. One is story motivated within the game and is defined as getting the best job in the game. This is the goal of the avatar. The second goal is getting the game complete screen, which is done by getting all the four main stats to 80% of their max value or above. In addition to this players can define their own goals within the game, like getting a better score than their friend, or getting all the four values to 100%. Having this kind of dynamic score system, lets the game stay fun and competitive even after players have beaten the game once. This should sufficiently provide competition within the game, which makes players strive to improve their tactics.

The result of having all of these elements in the game is that players experience the feeling Gee describes as empowered learners, which he argues is the most important thing for educational games to achieve.

6.5 The users

The main users of our game include developers, players and assignment creators. Each of these user groups will make use of a different portion of our implementation when used in connection to a course.

Developers

As our implementation is a proof of concept, it would be natural to include developers to tailor make assignment types for different courses the game can be used to supplement. Our assignments are based on knowledge related to the INF1050 course, and other courses might need other assignment types. These assignment types can easily be added with some entry level Java knowledge.

Players

The players will be the students attending the courses that make use of the game. They will only be presented with the main game view and will only need knowledge attainable from the course they are attending to play the game. The introduction given when they start the game should be sufficient for them to understand the gameplay, but a tutorial is also available to them.

Assignment Creators

The assignment creators will include the administration on the courses that make use of our game. They are given administration rights to the game, and are presented with additional functionality to create assignments for the game. This consists of an easy to use graphical user interface that requires no programming knowledge.

The players will be the most important user group for us, since our test will revolve around their

6.6 Summary

When designing our game, it was important that we chose a design that would fit our motivational criteria. In this chapter we gave an overview of the design aspects of our game, these include: in game currency, happiness, work experience, time and random events. These elements combined make up the challenge and entertainment value within the game. The gameplay consists of managing your allotted time to balance education level, work experience happiness and money while reacting to random events. The games educational content comes in the form of assignments related to the in game education. These currently include multiple choice (with or without images), draw-on-image, click-on-image and SQL assignments. We evaluated out design elements to include a sufficient amount of motivational aspects to be appealing to the end users.

Chapter 7
Implementation

In this chapter we will give a detailed overview of the implementation of the game. We will explain why we chose to use the MVC model and what benefits that gives us. Further we will look into the reasoning behind our class diversion and choices. We will specifically mention how each class was implemented and how it relates to the design of the game as described in chapter six. For a more detailed look on the code implementation, please review the source code in appendix D.

7.1 Overview

When using an object oriented approach to system development, as we chose for our project, it is important to identify all the objects pertaining to the project from a real world point of view. The core objects of the game consist of a player and the places the player can visit and interact with. In addition to that, certain places within the game have objects directly related to them, like work positions at a workplaces, and courses at the school. The game is designed with this general idea; that all objects in the implementation represent objects in the real world situation. This is in line with the guidelines of modern object oriented programming.

7.1.1 The MVC model

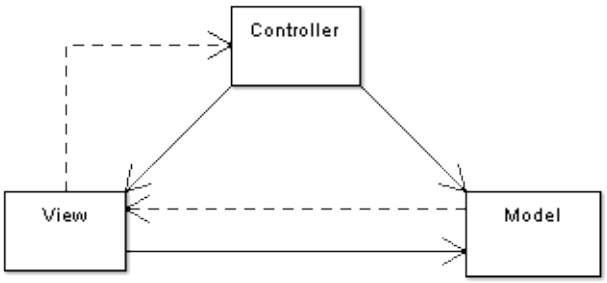


Illustration 15: The Model View Control model

In addition to identifying real world objects, we further divided the class implementation into a "*Model View Controller*"(MVC) («MVC», 2002) architectural pattern. When using MVC you divide the view from the model, so that our graphical representation of the model is not directly related to the model. The Control Layer is added to handle the all the information going in between the Model and View layers. The main benefit of using the MVC architectural pattern is that you can make changes to either model or view, without having to change the other. This is often used in large scale systems where changes to the view based on user feedback, change of user base or change of data structure in the core is highly likely. We chose the MVC model for simpler reasons. With two programmers on a tight schedule, the MVC architectural pattern let us work independently on different parts of the program for certain stages of development. With a standard for how data is handled in Model and computed in Control, parts of View could we written before the code related to it in Model or Control was written. This also simplified changes done during the course of the game development. We rewrote a large part of how data was stored in model when we decided to add some more complexity to how our assignments worked, but we could leave View unchanged as it all ready conformed to our chosen standard.

7.1.2 Brief walk through of how the system works

As the program is started, the Start method in Control is called. This method executes the code for loading the game places, courses and other necessary data in the game from their corresponding files. After this it calls the showAvatarMenu in view which lets the user create an avatar. As the user enters a name for their avatar, view returns the information of the players chosen name to Control. Control creates the avatar object, and connects it to one of the game places as a starting location (Small House as default). After this, Control calls upon View to create the main menu, and waits for user input. Throughout the game, from this point on, View will get the input from the user and send this input to Control. Control will then check if it is a legal action according to the data in Model, and inform View whether or not the action could be performed at this time. If the action could be performed, Control will edit the data in Model according to the action using the appropriate methods. Afterwards Control will call upon View to send feedback to the player. If the action could not be performed, Control will send a message to the player through View informing the player of why the action could not be done.

We decided that volatile storage of data as the game progresses was sufficient for this

program. All data relating to changes in values during gameplay is thus stored in memory. If you turn of the computer or shut down the program during play, all progress will be lost, but the game is designed to be completed in a single session. A save game feature could easily be added to the game by storing the values in the Avatar class to file, but was not deemed necessary for this version.

7.2 Class diagram

In illustration 16 we show how the entire system is connected. This picture clearly shows how the program is split into Model, View and Controller. From the image you can also see how they are connected through Control. As the system is fairly complex at first glance, we shall split this overview up into its parts and focus on one part of the system at a time.

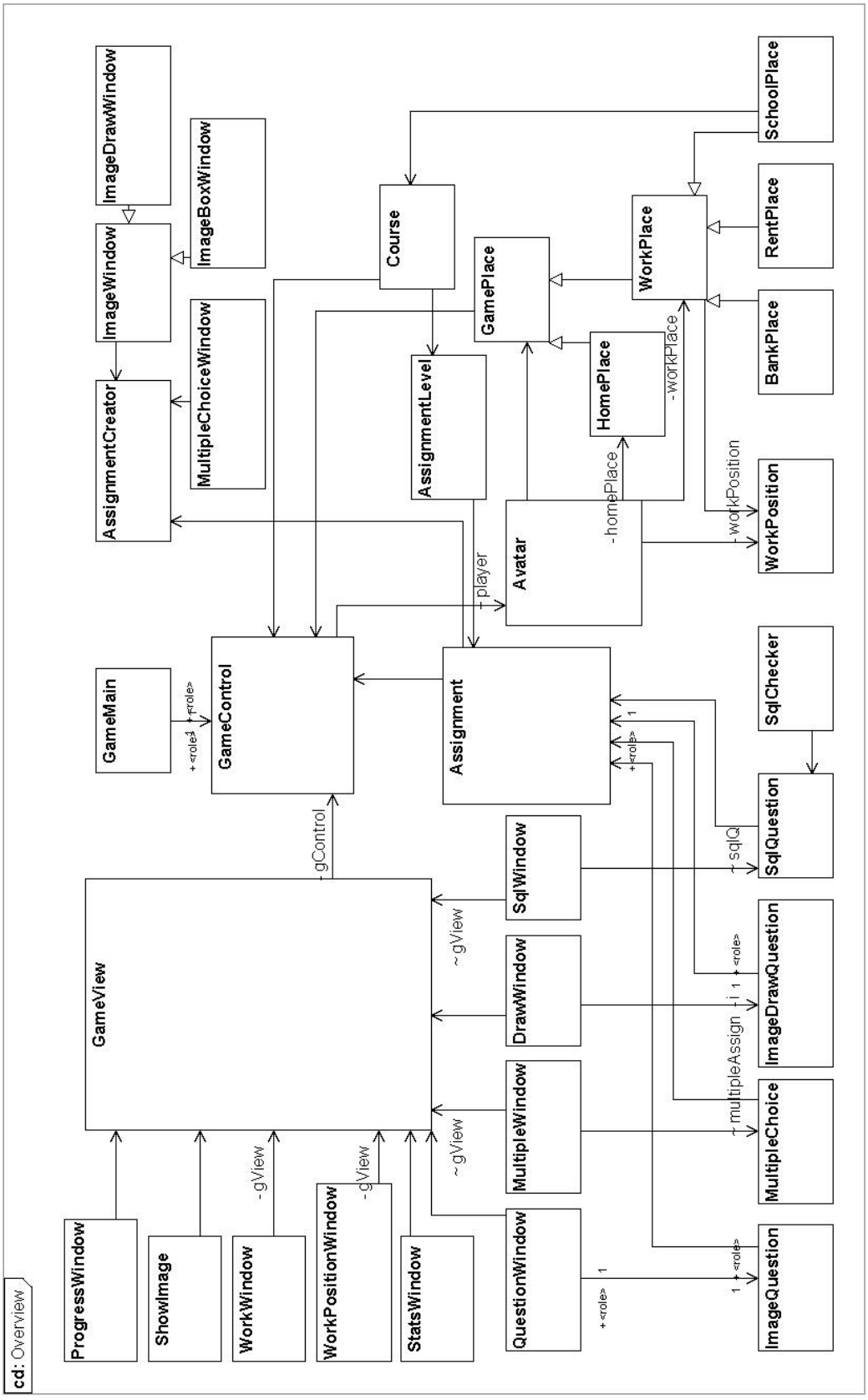


Illustration 16: Class diagram overview

7.3 Model

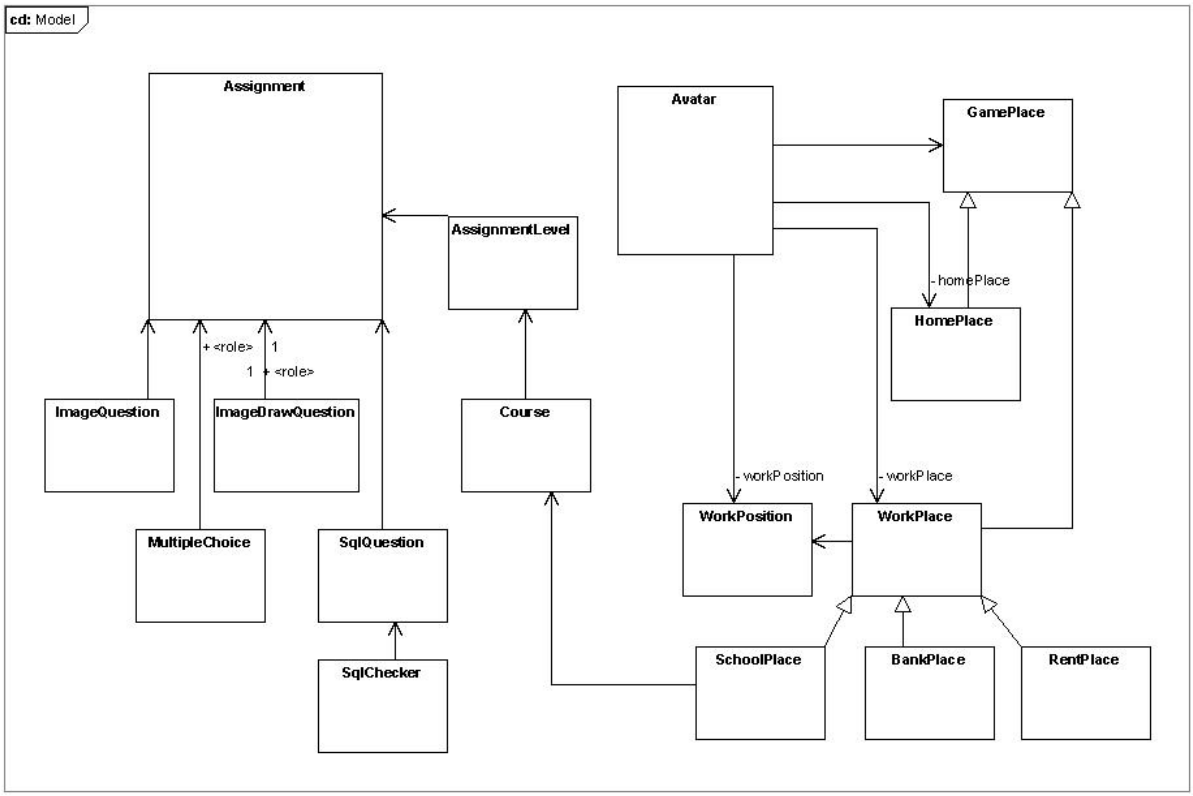


Illustration 17: Class diagram for Model

The Model takes care of all data storage as the game is running. Classes are created in model to represent all the different objects used in the game. Almost all of the classes are generated based on data from the file structure. This makes it easy to change values and names related to the objects without changing the code itself. This has several benefits. It can be used by administrators of the game to change for instance the difficulty of the game or change the whole setting of the game world by changing the in game names, without having to know Java.

7.3.1 Game Places

The GamePlace objects represents the places the avatar can travel to throughout the game. When the avatar is moved to a certain location he is given interaction options based on where he is located.

The game place structure uses inheritance as there is a lot of similarity between the different places. This is true to object oriented programming and simplifies communication between the layers as it is not always necessary to know what kind of GamePlace that is being

manipulated, simply that it is a GamePlace.

GamePlace

This is the standard GamePlace class. It is not used in itself but the other game places inherit code from this class. The GamePlace class contains code and methods pertaining to naming of the different game places. Having this standard class lets for instance View ask for the name of the current location of the player, without having to know what type of GamePlace the player is currently residing.

HomePlace

Inherits from the GamePlace class, and has additional functionality for handling rent and the initial cost of purchasing this home. The avatar is connected to a home place upon starting the game, and can change housing during the course of the game. When residing at a HomePlace the player has access to additional functionality, like resting. Small House and Large House are the names of the current HomePlaces.

WorkPlace

The WorkPlace class also inherits from the GamePlace class. All the work places within the game contain a HashMap of corresponding work positions, as well as code for getting information about available work positions. Every place in the game where the player can work, inherits code from this class. This code in this class is used for two major purposes, working and applying for work. When players are located at the WorkPlace they currently hold a job, they are given the option to spend time working, and code from the WorkPlace class is used to get information from the work position such as salary.

BankPlace

Inherits from the WorkPlace class. In addition to the functionality of the WorkPlace class, it has information about how much money the player has stored in the bank and functionality for withdrawing and depositing money.

RentPlace

Inherits from the WorkPlace class. It handles information on the amount of rent a player has to pay, and when it is due.

SchoolPlace

Inherits from the WorkPlace class. The SchoolPlace class contains a HashMap of available courses the player can take, as well as code for getting the appropriate CourseLevel for the Course. The data for the Course objects are loaded from file upon starting the program.

7.3.2 Courses and Assignments

The Course classes are a central part of the educational element of the game. These classes hold all the information about the classes the player can take within the game. There are currently three different courses. Each course has a list of assignmentLevels and each assignmentLevel can contain one or more assignments. This is what makes up the structure of the course. For every education level the players want to achieve with a certain course, they have to either answer one random question from the assignmentLevel or every question in that assignmentLevel. This depends on whether the assignmentLevel is classified as an exam or not. Each assignment within an assignmentLevel can be either one of the question types implemented.

All of this gives a lot of possibilities when creating a course structure. You can vary the amount of random question on a level, decide to not have questions on certain levels, decide which levels to add exams and how many assignments these exams should contain.

This assignment structure is loaded into memory in its entirety upon starting the game.

Course

The Course class contains information about which assignment level that is given at the player's current education level related to the course. As the player chooses to take a course, the course object loads the correct assignment level from the HashMap using the players education level as the key. This assignment level is then presented to the player based on the values set for this level.

AssignmentLevel

The assignment level has a HashMap linked to all the assignments that are stored for this particular AssignmentLevel, and a Boolean value for whether this is assignment level should be treated as an exam or not. If the assignment level is an exam, all the assignments for this level is presented to the player in turn, and the player has to complete a set amount of the assignments to pass. If the exam Boolean is set to false, a single random assignment from this

assignment level is presented.

Assignment

This is the standard assignment class from which all the other assignment classes inherit code. It contains basic information about assignments, such as assignment name, corresponding AssignmentLevel, question and answer; all the common denominators for the different assignment classes.

ImageQuestion

The ImageQuestion class inherits all the basic code from the Assignment class, in addition to specific code to deal with click-on-image assignments. This includes information about the image placement and overwriting the question and answer to be coordinates on an image.

ImageDrawQuestion

This class is similar to the ImageQuestion class, but the answer and question are overridden to contain more than one set of coordinates. This allows us to make assignments in the form of drawing an answer on an image.

MultipleChoice

MultipleChoice inherits from assignment, and has in addition to the normal assignment information, an array of the choices that should be listed as answer alternatives. MultipleChoice assignments have the option of having a corresponding image related to it.

SqlQuestion

This question type, and its corresponding class SqlChecker is no longer in use in the current implementation of the game since the INF1050 curriculum no longer includes SQL on a level where SQL parsing of answers where necessary. The SqlQuestion class contains a set of standard database files related to the question. It also contains code for creating an object structure of an SQL query and checking the object structure of the correct query against the query given by the player.

7.3.3 Avatar

Since the avatar is the most important object in the game, it is only natural that the Avatar class is the core of the model. The Avatar class contains all the information pertaining to the player, including happiness, money, educational levels and work experience. The Avatar class

is connected to a HomePlace at start up, also has a link to a WorkPlace and a WorkPosition as soon as the player gets a job in the game. Every time the player tries to perform an action control checks against the values stored in Avatar to see if the player is allowed to make the move. Almost all of the values that represent progress in the game are stored in the Avatar class.

Integers	**Explanation**
time	time left this turn
week	corresponds to the number of turns used
money	how much money the player has on him, does not include bank balance
happiness	represents the happiness of the player, the game ends if this value gets to low
workExperience	value corresponding to the level of prestige for the best job the player has had
educationLevel	one value for each course. increases as the player completes assignments
clothesQuality	type of clothing currently owned
clothesDuration	how much time left until new clothes are needed

Table 3: Key values stored in the Avatar class

In addition to these values, the Avatar class stores the player name, and has several object pointers. The object pointers link the avatar class to WorkPlace, WorkPosition and HomePlace. The WorkPlace pointer is used to get information on the players current WorkPlace, this is for instance used when moving around in the game to see if the player can work at his current location. Similarly the WorkPosition pointer is used store information about which position the player has at the current WorkPlace. The HomePlace pointer represents the player's current home location. At this location the player can relax, eat home cooked meals, and each turn starts and ends at this location.

7.3.4 WorkPosition

The WorkPosition class represents work positions in the game. Every WorkPlace has a list of WorkPositions. These WorkPositions contain information about the salary of the job in question, and also the requirements for getting the job. These requirements include

educational level, work experience and clothing level. The information on the WorkPositions is the key part to pertaining a work hierarchy in the game. The balancing of these levels is what controls the progress in the game. As a player gets increased levels of work experience, and wants to apply for a better job, the requirements set in the WorkPosition object ensures that the player is encouraged to use the different features in the game like the clothing store and education. A list of some of the more important values is shown below.

Value	**Explanation**
name	The name of the work position. Same as presented to the player when applying for work
educationLevel	The education level needed to get the job. One for each course in the game
salary	The amount of money gained from working one time slot with this work position
workExperienceGained	The work experience you get from working at this work position
workExperienceNeeded	The work experience needed to get this job
timeToObtainWorkExperience	How long you have to work at the current location to obtain the work experience it gives
clothesQualityNeeded	The level of clothing quality needed to get and work at this work position

Table 4: Key values in WorkPosition

7.4 Controller

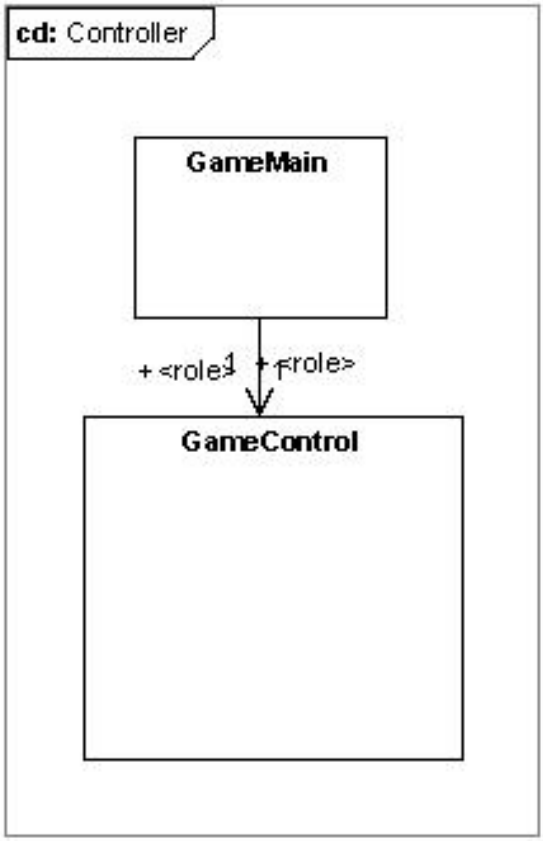


Illustration 18: Class diagram for Controller

Controller consists of two classes, a simple main class which starts the program and the GameControl class where most of the calculation and checks are performed.

7.4.1 GameMain

GameMain is the standard java main class. It initializes GameControl, and runs its start method. This ensures that there is no need for static classes and is a basis for object oriented programming in Java.

7.4.2 GameControl

GameControl is the main class of the game. It is responsible for all data flow between view and model, and it also handles all the important data checks and decides what the player can and can not do. GameMain makes sure that the start method in GameControl is called as the program is started, and this method initializes the View and builds the model from the data in

the file structure. After the data structure is created and the main window is initialized, GameControl waits for user input through View. As all the important methods relating to gameplay reside in GameControl, we will list some of the more important ones here.

Method	Explanation
start()	loads the data from the file structure and calls on View to initialize the main window
newAvatar(String name)	creates the avatar object after the player has selected a name
loadGamePlaces()	loads the gamePlaces from the file structure. Called from start()
loadSchoolCourses()	loads the courses from the file structure and creates course and courseLevel objects. Called from start()
loadAssignmentLevels(Course newCourse)	creates assignments objects for a courseLevel after the course has been created.
correctAssignment(Assignment assignment, String answer)	corrects assignments based on user input. This method is overloaded for different assignments types

Table 5: Important methods in GameControl

In addition to these main methods GameContol has lots of get-methods to facilitate getting information from Model to View (i.e. getBankBalance()), and methods checking every action the player takes against values in model (i.e. applyForWork(String work)). For a full list of methods in GameControl see Appendix D.

7.5 View

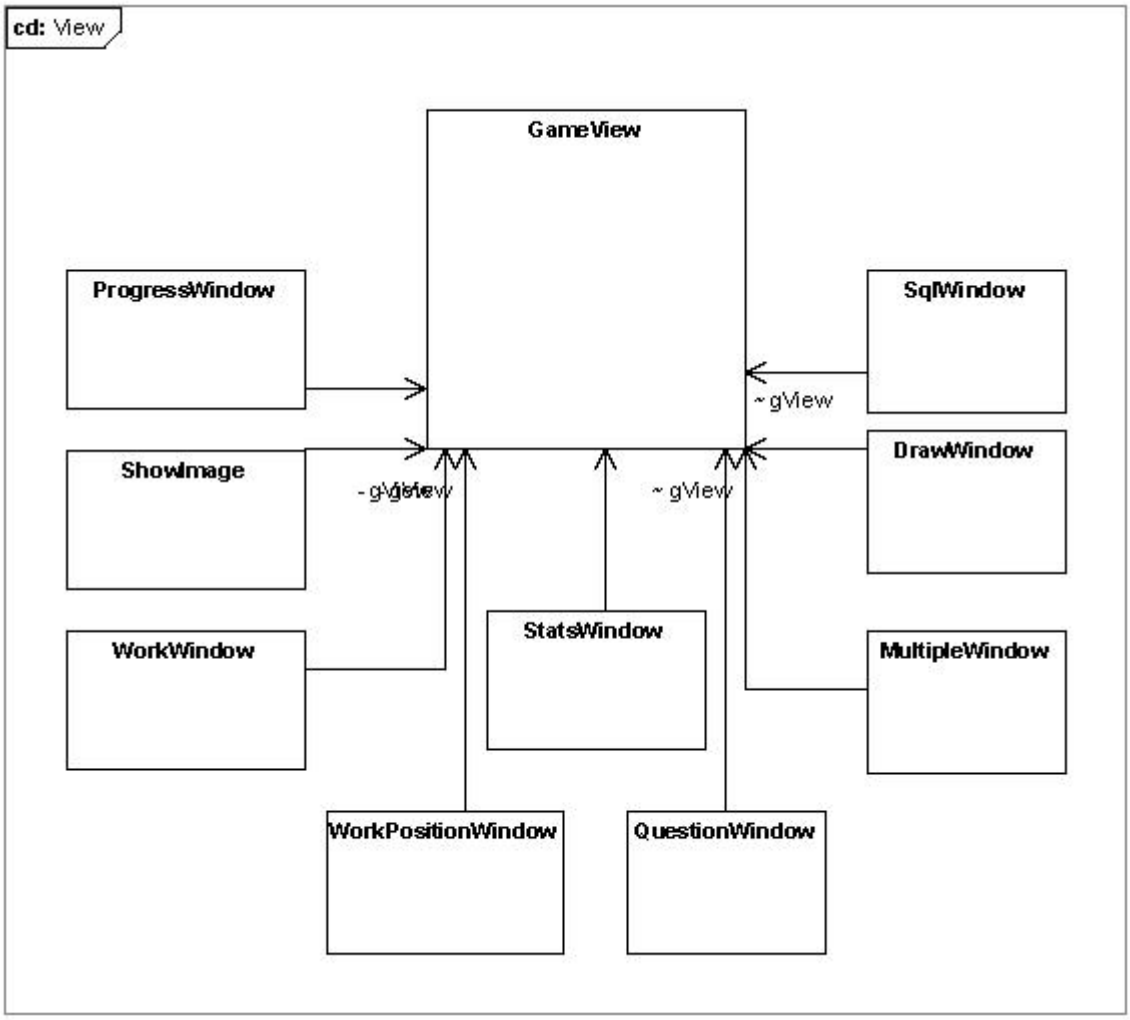


Illustration 19: Class diagram for View

The View handles all the input from the user, sends it to control for processing, and show the corresponding feedback coming from control. The View is comprised of a main window representing an overhead view of a board game, and several smaller windows for the assignments, work application and user feedback. Since the View is mainly a tool used to facilitate input and output of the game, we will only go over it briefly.

7.5.1 The Main View

GameView

The GameView class is used to layout the main view as the program is started. It handles the process of user log in, and the drawing of the game board with a turn timer and user statistics.

The ActionListeners related to GameView handles all the basic functions of sending feedback from the users to Control. Among other things this is used to manage user movement through the game. GameView consists of many layers of panels and layout managers. Each of these panels contain differents elements that make up the main window of the game. The panels included in the main window are: MainMenu, CenterMenu, Player Stats, Progress Stats, Log and Clock.

MainMenu

The mainMenu panel contains all the buttons representing the places in the game. These are the buttons the player uses to move around within the game world. These buttons have icons on them that change based on the player's current location. They are represented by a dark circle if the player is not at that location and a smiley at the players current location. This smiley changes when the player changes clothing.

CenterMenu

The centerMenu is located in the middle of the screen and changes based on the location of the player. The centerMenu contains buttons representing all the actions the player can perform at this location. This menu is created every time the player changes location, since the action the player can do at the location changes during the course of the game. For instance if a player works at the location he is at currently, a work button is added to the centerMenu.

Player Stats

This is the small panel at the top left of the screen. It holds the most important information the player needs each turn such as time left for this turn, money and bank balance.

Progress

The progress panel is located under the player stats panel and contains a graphical representation of the four progress values in the game: Money, Happiness, Work Experience and Education level. These values are presented as sliders representing how close the player is to achieving the end goal of the game. When one of the values reaches the limit needed to complete the game, the icon over the slider light up to inform the player that this value has reached a sufficient level.

Log

The log is located at the bottom of the screen and is continuously updated with text describing every action the player has taken, and the results of these actions. The log is meant to be used as a way for the player to keep track of what he has or has not done in the past turns. This is a help full tool which the player can use to plan each turn.

Clock

The clock panel has a graphical representation of the time left this turn. Since so much of the choices the player has to take are based on how much time that is left, it was important for us to include a large visible representation of this. This clock lets the player check the time with a simple glance, rather than forcing them to look at a small numerical value. As time is spent during a round parts of the clock change to a red colour to indicate how many time units have been spent.

7.5.2 Assignment Windows

Whenever a player takes a course in the game he is presented with an assignment, or several if it is an exam. The assignment windows handle the input related to these assignments. Since the different forms of assignments asks for completely different forms of input from the user, these windows are custom made for each assignment type. Specific info on the implementation of the assignments is handled in our overview of Model.

ImageQuestionWindow

The ImageQuestionWindow class presents the user with the question related to the assignment and the image. The point where the user clicks on the image is sent to Control.

MultipleWindow

This class handles the multiple choice assignments, with or without corresponding images. The user is presented with a question, and a list of possible answers as a list of radio buttons. Once an answer is chosen, the user confirms by pressing on a button labeled "OK". The answer from the selected radio button is then sent to control for correction.

DrawWindow

The DrawWindow class handles assignments of the "Draw On Image" type. It presents the user with a question, a corresponding image, and lets the user draw on the image to complete the assignment. The user has the option to clear the image of what is drawn on at any time,

and can thus redraw on the image if an error was made. Once the user is satisfied that the drawn line on the image is correct, the assignment is confirmed with an ok-button.

SqlWindow

The SqlWindow class handles questions related to SQL statements. It shows an arbitrary number of tables representing database files, and asks the user to present data from the database in a certain fashion using standard SQL statements. The input from the user is sent to the SQL parser.

7.5.3 General Purpose Windows

StatsWindow

Shows useful information on user statistics like money, and weeks until rent is due. The window is accessed through a drop-down menu in the main window.

ProgressWindow

This window shows how far the user has progressed towards the end goal. The information is presented as vertical sliders, where the top of the slider represents the goal for the particular game value. The values shown in this window are: happiness, money, work experience and educational level. These are the main characteristics that determine your success within the game.

ShowImageWindow

A general purpose window that lets Control present the user with images. Useful for giving the user image based feedback and information.

WorkWindow

Lists the places where the user can apply for work in the work application process.

WorkPositonWindow

Lists the work positions available at the chosen work place in the work application process.

7.5.4 Assignment Creator

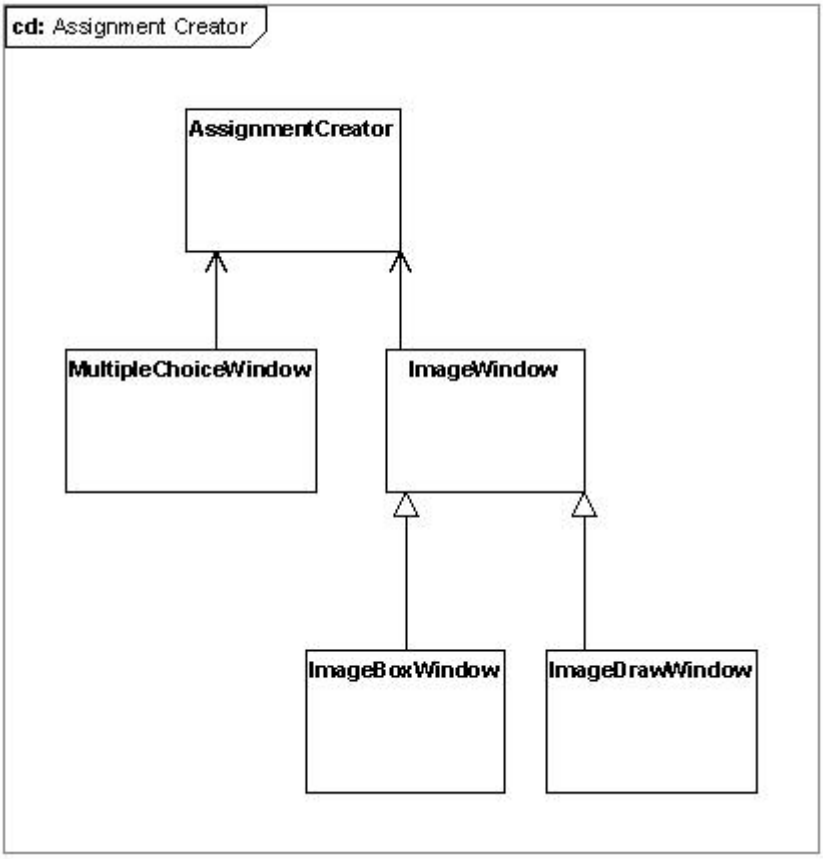


Illustration 20: Class diagram for Assignment Creator

The AssignmentCreator class is an added module to View which lets users with administrator rights create assignments through a simple user interface. The assignment creator is accessed through the admin menu and opens up in a new window. It lets the user edit old assignments and create new ones, and chose where the course and courseLevel for the assignment. This can be used to easily redesign the assignment structure within the game. The AssignmentCreator is essential for scenarios where for instance group teachers add content to the game on a regular basis. Based on the type of assignment the user wants to create or edit, the AssignmentCreator presents the user with different user interfaces:

Multiple-Choice Question

When creating multiple choice questions the user is given the option to relate an image with the question. If an image is wanted, a file chooser opens up, and the user can choose a file using an interface similar to window explorer. The file is copied to the game folder, and linked to the question. The basics of a multiple choice question are created by typing in a question, and a desired amount of alternative answers. One of the answers is then checked as

being the correct one.

Click-on-Image Question

When creating Click-on-Image questions the user is first asked to choose an image in the same way as for multiple-choice questions. They are then presented with a window containing the image and a box used to type in the question. The user then makes a selection box over the image representing the correct portion of the image to click on. This selection box is clearly marked on the image by a red square, and the user can redo the selection until satisfied. Once the user if happy with the results they confirm the creation of the question with a button labeled "Ok".

Draw-on-Image Question

Draw-on-Image questions work in a similar way to Click-on-Image questions. In this version of the game, Draw-on-Image questions are created by selecting two boxes on the image. These boxes represent where the player has to start and stop drawing. This type of assignment could be made more advanced by creating a more flexible way of defining the constraints on where the user can draw on an image in order to get a correct answer, but the simplicity of the current implementation makes it a lot easier to create Draw-on-Image questions.

SQL question

The AssignmentCreator does not support a user interface for creating SQL-questions as the question type was removed from the current version of the program before the AssignmentCreator was implemented. SQL questions can still be easily created through a simple text based interface.

7.6 File system

7.6.1 File storage vs. database storage

When making the choice of how to store the files related to the game, there are two main options either data-base storage or text based file storage. Database storage has several benefits over file storage, such as access speed and multi-user access. We chose to use simple file based storage however.

Database storage provides a lot of advanced features like data consistency through secure transactions and faster access times for common queries. But none of the key features of

database storage were deemed necessary. The files are only read upon starting the game, and loaded into memory. The faster access times database storage could provide here would we unnoticeable.

The content of the files are only changed when editing assignments, adding new assignments or changing game files in order to change the game setting or difficulty. All of these actions are meant to be performed by a single user with administrative rights, and most likely not more often than on a weekly basis. This alleviates the need for user access control and secures transactions.

Database storage also requires some overhead as you need a database API. This would increase the size of the program with by a significant amount without any noteworthy gains.

However our main reason for not choosing database storage is the fact that maintaining and editing a database requires the users to know SQL to access the data. We can probably assume that the administrators of the INF1050 course know SQL, but if the program was to be ported to any other courses, it would be much less likely that the administrative users know SQL.

File storage is a lot easier to implement and maintain. Since one of the requirements of our system includes that end users should be able to create and edit assignments on their own, it was vital for us to have a file system that didn't require a lot of prior knowledge to maintain. Text based file storage has a huge benefit that everyone is expected to know how to edit, create and manage simple text based files at a university level. This is something that all students learn at a high school level now.

As the choice fell on text based file storage, we were left with the choice how the data within these files should be stored. We could use a standard like *Extensible Markup Language* (XML)(«XML», 2008) or create our own standard for the game. The XML standard is the by far most common standard related to informational based text storage. XML was originally created to handle large-scale electronic publishing, and as such has a structure that is far more advanced than what is needed for our project.

Even though XML is the most used standard for text based data storage, it is still not something that you can expect end users to know. The standard in itself is relatively easy to learn, but we can not expect everyone set to administrate our game to do so.

In the end we decided on a simple list based information structure for our data files. This structure is as intuitive as reading plain text. It has headers and information following this header. New headers define new information blocks. Much like how this thesis, or any text, is structured. This is something we can expect anyone to understand intuitively.

Another advantage of this simple structure is that the text files themselves become smaller. This saves on storage space and access time.

7.6.2 The directory structure

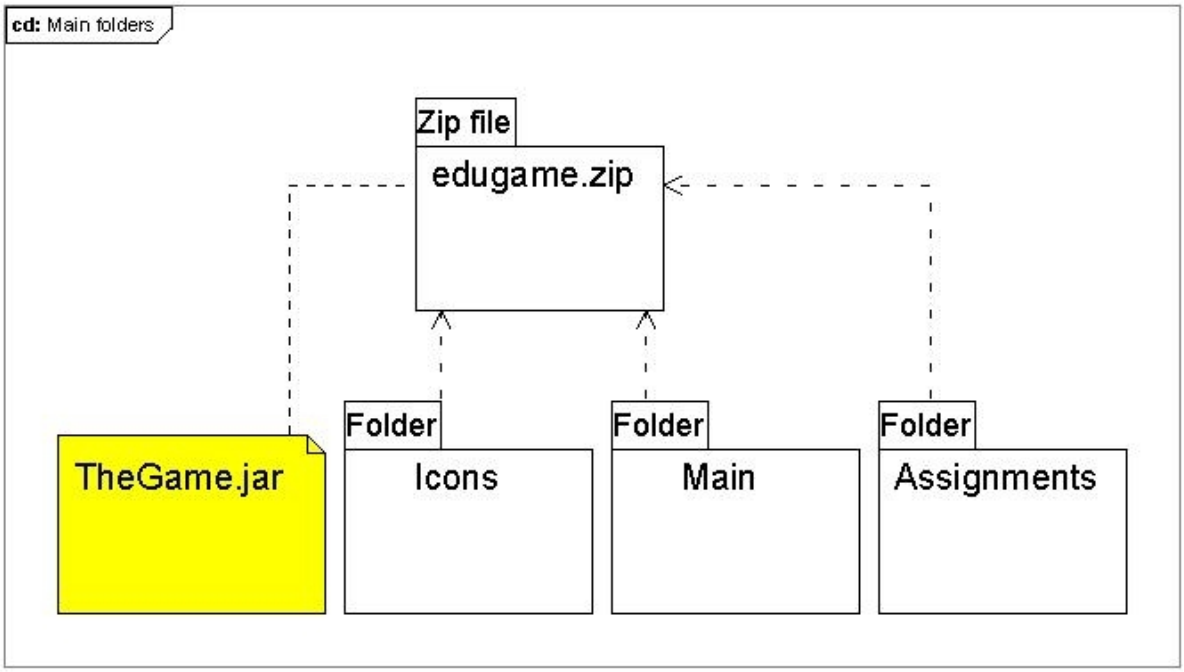


Illustration 21: Main folders and JAR

The game has four main directories: the code files contained in a jar file («JAR», 2008), the icons folder, the main folder and an assignment folder.

Code files

The code files are contained within an executable Java Archive (jar) file. Jar files are used to bundle multiple java files within a project into a single file. Even though most OS' view jar files as a single file, they contain a directory structure within themselves. The main advantage of packaging projects into single jar files is that it makes the program executable with a single operation (double click in windows and run file.jar in linux).

The jar file for our game has three main directories directly linked to the three packages of the

program: Model, View and Controller. This is in line with the standards for creating jar files. Within these directories are the files related to each package. For instance the View directory contains the java files related to the different window classes.

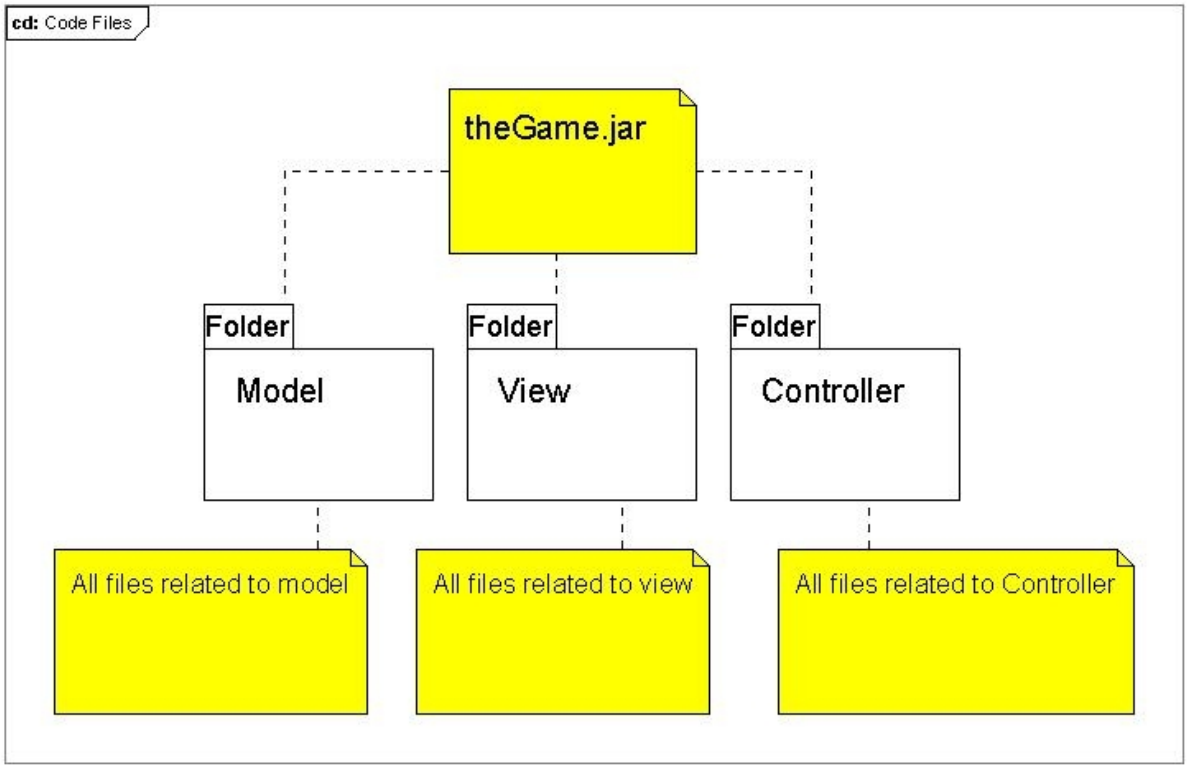


Illustration 22: Code files

Game files

The game files are stored in a folder named Main. These files are related to all the in game values that involve gameplay, balance, difficulty level, naming and story. These files are meant to be easily accessible by administrators so they can tweak the gameplay to their liking.

Assignment files

The folder labeled Assignments contain the assignment hierarchy within the game. The folder is divided into three sub-folders named course1, course2 and course3. The naming of these three courses in the game are controlled via the files in the Main folder. In each of these folders are files labeled with a number and a file extension that is either .ass or .exam. The numbers correlate to the assignment level of the assignments contained in the file, and the file extension decides whether the assignments should be treated as an exam or single/random assignment.

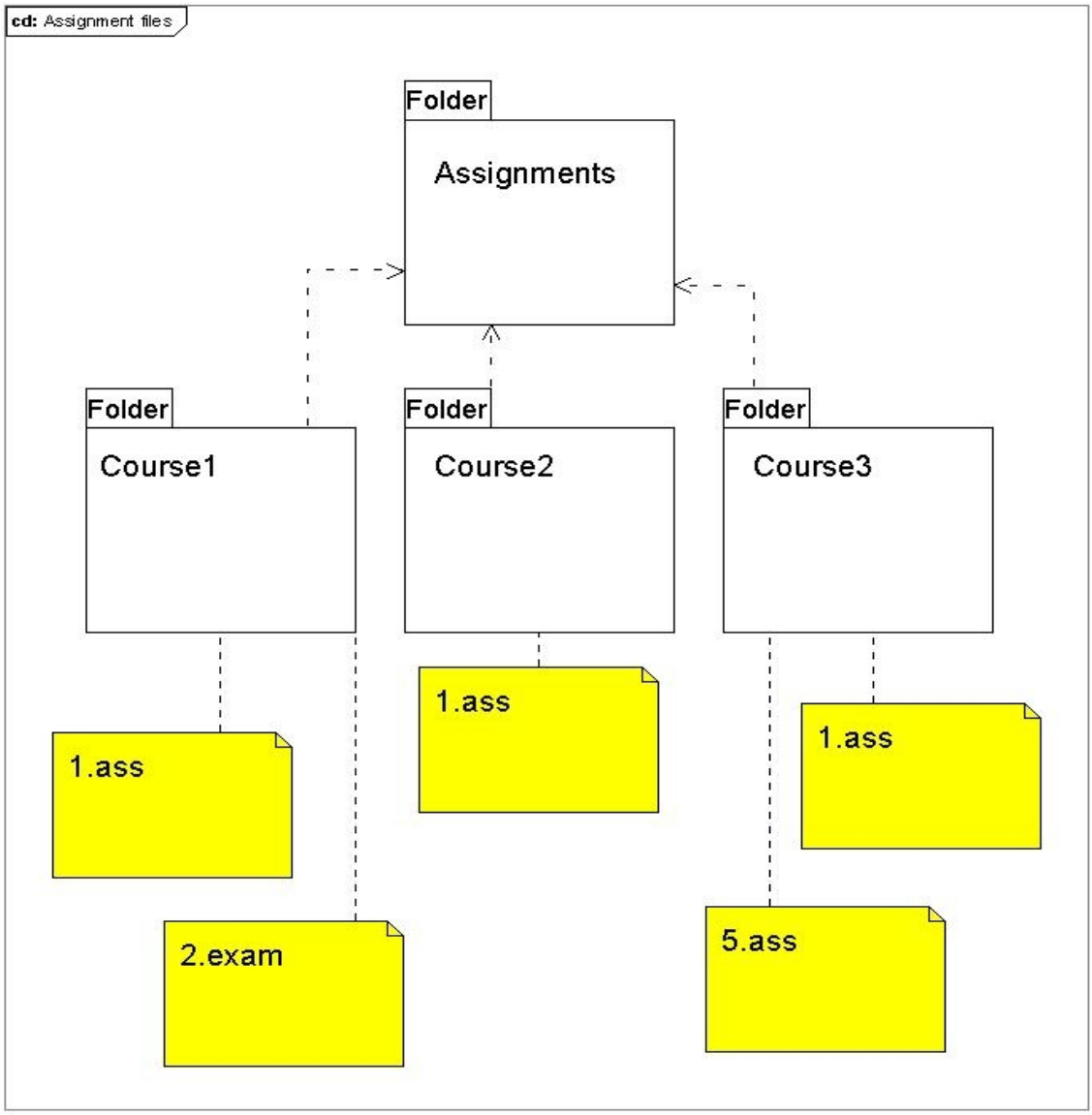


Illustration 23: Assignmenst file structure

Icon files

The Icon folder is a single folder containing the images related to View. This folder contains clock-, avatar- and progress-icons. These images are used in the GUI. The clock icons are images representing the ingame clock, the avatar icons are used to illustrate the different kinds of clothing the avatar can use and the progress icons are used to highlight the four different goals when they are reached.

7.7 Summary

In this chapter we went through the basics of our implementation. We explained why we chose the MVC model to ease our development process and to keep our code structured. We described how the game implementation works with a detailed look on what happens on start-up. The class structure of the implementation was presented in detail to give an understanding on how the different classes are related to each other. Some of the core classes where explained in detail to give more insight on design elements. We went into detail on why we chose file based storage over database storage. This was mainly done to make it easier for administrators of the game to manage and create assignments. The last part of the chapter gave a structured view on the file system the game uses.

Chapter 8
Testing

In this chapter we shall give a detailed description of how we performed the testing of our game. We split the tests into two rounds. The first test having the purpose of preparing our program for the second test. The second test is the most important one, as it is the one linked to finding the answers to our thesis.

8.1 Preliminary Testing

To better prepare for our final testing of the program on the INF1050 students, we decided to have a preliminary test on a small group of people. The main purpose of this test round was to see how users interact with the interface of the game, what parts of the game are understandable/unclear and what they think about the potential for the game. Our hope was that testing on a small group of people before the final testing would make our game better, and hopefully we would find and correct small bugs we have overlooked.

8.1.1 Goal

The purpose of this test was not to gauge the learning potential of the game, but rather we focused on the interface and ease of use. As the test subjects explored the game we focused on their immediate reactions to the interface and game mechanics, trying to figure out what confused them and what elements of the interface that lead them to take certain actions within the game. The main purpose of this was to find out what parts of the interface that needs improvement before our final tests. Secondarily we focused on the game-play aspects of the game, and how they affected the general view of the game, giving us valuable input on how to balance the difficulty and progress. Whenever the test subjects encountered any of the learning aspects of the game, we had either made sure the problems where trivial, or we gave them the answer, so that they could focus on the interface related to the learning aspects rather

than the learning aspect itself. The feedback from these tests will hopefully give us enough information to make the game-play and interface enjoyable, so that we can focus on the learning aspects for the final series of tests.

8.1.2 Evaluation technique

When we looked at the goals we had set for this first preliminary test, we found out that we needed to be in the same room as the testers to be able to pick up everything the testers actually meant and felt about our program.

8.1.3 Think aloud evaluation

Think aloud evaluation is a testing method where the tester and the person giving the test (test head) are sitting in the same room. After a brief introduction to the program by the test head, the tester is allowed to test the program at their own leisure. The tester is encouraged to tell their thoughts about the program aloud while examining the program. The test head can then sit and evaluate the progress of the tester, while at the same time picking up the testers comments and emotional body language. A good thing with think aloud testing is that the test head can easily guide the tester with comments and tips if the tester is stuck or pondering about something.

8.1.4 How the test was conducted

The test subjects were given a document that briefly described our master thesis and the purpose of this test. During the whole test procedure, we had one interviewer focused on interacting with the test subject, either to make sure they tested the right elements of the game, or to answer any questions. The other interviewer would take notes. After having read through the entire document explaining our project and their tasks, the test subjects were presented with the program, which they where free to explore on their own terms at first. Our hope was that the in-game help and explanations would be enough to guide them through the game, which largely seemed to hold true. As they tested the program we asked them to verbally describe what they where doing and why, and give general feedback on the game. After a short while of free form testing, we guided the subjects towards the tasks described on their info sheet. These tasks where designed to ensure that the test subjects explored all the different parts of the interface.

Test subjects

We performed the test on two persons with different backgrounds. We chose one master student from IFI and the other one is just finished with his bachelor degree from MatNat.

Tester1:
Gender: Male
Age: 25
Degree: Just finished bachelor program at MatNat (UiO)
Prior knowledge related to our field of work: Taken the first basic computer programming course (INF1000) lectured at UiO. Know the basics about how a computer is used.

Tester2:
Gender: Male
Age: 24
Degree: Currently writing his master thesis at IFI, UiO
Prior knowledge related to our field of work: Played a lot of computer games, good at computer programming, taken several computer courses at UiO.

The master student told us he had good prior knowledge about programming and design of user interfaces. He has taken INF5270 which is a course much concerned with user interfaces and usability of computer programs and web pages. The bachelor student on the other hand had very little education about the things we are testing, but he told us that he had used a lot of different computer programs in his education. By using two test persons with such different backgrounds we hoped to get two different views on our program, and pick up different thoughts and aspects from the testers on our game.

The text that was given to the testers can be viewed in appendix B.

8.1.5 Results

When first presented with the game, both testers were curious on what the different elements in the user interface did. They both created an avatar with a name without problems, and entered the real game. The first problem both testers encountered was to find out what the

different location names meant. They quickly understood that it was a central issue to manage their time, but they had a hard time understanding where to move to perform certain tasks. For instance tester2 said: "Where do I start?" (Translated to English by authors) When given a tip that each place-button had a mouse drag-over tool-tip with a description on what action one can perform at the different places they seemed to understand a bit more. Still they both thought that the tool-tip was a bit slow and tiresome to use to find a specific location. After some minutes both testers understood the basics on how to move the avatar around the game, and where the action buttons for the places popped up etc. After a while, when they had gotten used to the user interface, they both liked the simplicity of it. Tester2 said: "It is not fancy or anything, but the issue here is the content, not the graphical."

Both testers also pointed out that there was no information at start-up on when the rent for the hired apartment was due, what it cost and where to pay it.

The mini games were well implemented according to both the testers. An issue tester1 mentioned was on the consistency in the layout of the questions. He meant that the questions always should come at the upper part of the question window, and again they lacked some prior information on how to answer on some of the questions. The multiple-choice question should be clearer on what it is that marked the right answer, and the drawing-question could use some more information on how to actually draw. Even with these comments, they easily managed to understand what to do and complete the questions in a good way. They also felt they lacked some feedback on how they did in the questions.

When it came to applying for work, both testers were very confused about where to start the application process. Tester2 got rejected from 4 jobs, and said "There should be a way to better understand what jobs one can get. They should be arranged after difficulty or something". At one point Tester1 also got the message that he should become better at System Development in order to obtain the job, and asked if he needed just one more level or several and asked where he could obtain that information. Another issue they both complained about the during job application process was that if one failed to qualify to a job one jumped two sub-menus back instead of one.

During the gameplay the testers felt that they lacked a nice overview on their total progression and other small information windows on things like where they worked, what their current salary was etc. Tester2 also meant that every time an avatar stat changed during gameplay it

should be listed in the bottom log window. In general both the testers felt that there was little information on what they where supposed to do, and information about the different elements in the user interface.

When it came to playability it wasn't surprising to hear that they both felt the game to be boring and it lacked variation. For this testing there where just a few educational mini games implemented, and they felt they where just wandering around without a goal. Both testers pointed out several bugs that we were aware of, but these were mostly related to gameplay issues that we had not finished implementing yet at the time of the testing. Another major problem that we saw clearly throughout the testing was the issue on the difficulty balancing time, cost and income.

8.1.6 Evaluation

We noticed that we need to give a lot more information to the players. When we as developers play the game, we know a lot of the background mechanics and automatically play the game a certain way. The testers had no idea about the background mechanics and played a very different way, and this showed us many potential problems that can occur during gameplay.

We need to add a more thorough introduction screen with the different elements in the user interface and about the gameplay in general. This could be added in form of a long explaining text or as a tutorial with images. Furthermore we need to change the names on several of the locations in the game in order for the name to represent what kind of place it is in a better way.

The mini games were understood well and liked by both the testers. We also showed them how easy it was to add new questions to the program, and they really liked this feature. Still we need to do some small adjustments to give all the questions the same layout in order to preserve the consistency though out the gameplay.

The work application process really needs to be looked more into. We need to make it more intuitive for the players to understand which jobs they gave a chance at getting. The job application process should not be a guessing game as it is now.

All in all we need to add much more information about several small things in the game. The testers mentioned a lot of issues we already were aware of, but also pointed out several others

that needs to be fixed before we test the program on a larger user group.

8.2 Revision of the game

In the second round of testing we tested the game on a larger user-group. We made some changes to the game from our preliminary testing, and added a tutorial that explained the most basic parts of our game. The tutorial can be viewed in appendix A. The biggest changes we did was reworking the career ladder, making the assignment structure more advanced and adding tons of textual feedback messages.

The career ladder was redone in order for the flow of the game to be more natural. In our first implementation the connection between job title, salary and requirements were inconsistent. By balancing this it became easier for the players to understand what jobs they could apply for. We also added better information on what needed to be improved in order to get certain jobs. This was mainly done by creating a dynamic feedback feature to the job application process and informing the players when certain stats had increased sufficiently to be suitable for new jobs. The dynamic feedback feature gives the players an indication of what stats they need to improve and how close they are to getting the job. We did not want to present these values directly as we wanted to maintain informational complexity and uncertainty in the game.

Illustration 24: Job application window

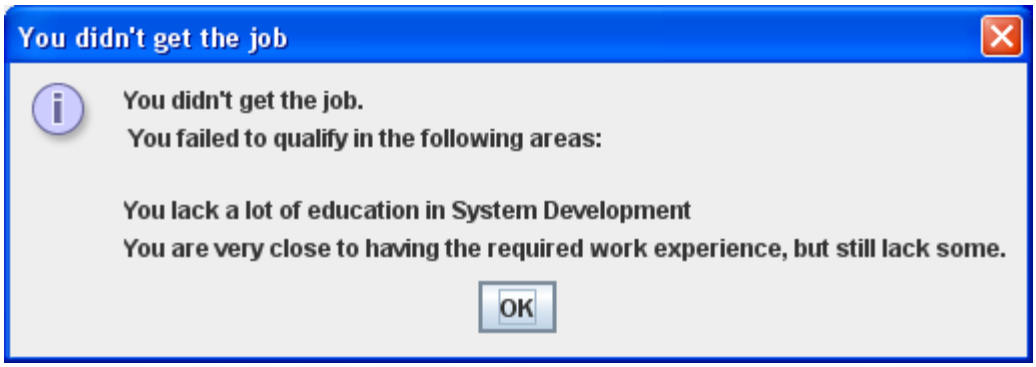


Illustration 25: Feedback on job application

The assignment structure was also redone. We wanted more flexibility in the way we could present assignments to the players. The new structure gave us complete flexibility in the number of assignments per level, if it should be treated as an exam or not and made the assignments more consistent. Reworking the structure also made it easier to edit and add assignments. This helped us a lot when we made new assignments for the final test round.

The gameplay was made a lot more intuitive by providing the player with more information. The log was changed to give detailed information on every action the player takes. In addition to this we made pop-up windows with all the important notices that the players need to know in order to plan their turn. A part of this change was remodelling the weekend reports that come at the end of each turn. The weekend report now formats and presents a detailed description of every event that is scheduled to happen during the next turn. This includes when the rent is due and when the player needs to purchase new clothing.

Illustration 26: Example of an informational pop up window

We also wanted to make the goals a more integral part of the game. The version of the game used during the first test round did not have any clear way of completing the game. We wanted to see how the gameplay functioned for people who did not know the game mechanics before we choose the goals. We decided that the game needed concise goals, and different

goals that the players could strive to achieve depending on their skill. We decided to add a story related goal of getting the best job in the game, as well as additional goals for managing the four main player statistics in the game (Work experience, happiness, money and education).

We implemented a score keeping system that tracked player progress, time spent and commodities gathered. The primary influence of the end score is how fast the player is able to increase the main statistics. We linked several different goals to these statistics. The easiest goal to strive for is just improving one's best score. The player can end the game at any time, or the game ends because the player failed. At this time a score is presented to the players based on their performance. Additionally there is an end goal consisting of getting all the four statistics to within a certain percentage of their max value. Achieving this presents the players with a congratulatory screen and an indication of their success on the main screen. The last and hardest goal the players can strive for is to get perfect scores in the shortest amount of time.

We also changed all the names of the different places in the game. The original game were made with humour in mind and as a reference to the original Jones in the Fast Lane. For instance the clothing store was called Clothes'R'us, a play on the American toy store company "Toys'R'us". Even though the testers found the names to be humorous, they also caused some confusion. For the sake of simplicity we decided to change all the names to descriptive names that clearly indicate the purpose of the location to the player.

The general game balance was also redone. We saw that the game was too hard for players who were unaware of the game mechanics. Our experience of the gameplay was completely different than what our testers experienced. We made it easier to climb the career ladder, we gave the players more time each turn, a lot of actions now cost less time points to perform and we made it easier for the players to get enough money to pay for rent and other expenses.

Some additional functionality was introduced to improve the gameplay, like rent income from money deposited at the bank and lottery where the player can win money.

The user interface was changed to accommodate all the above mentioned changes and also based on the feedback from the first test round. Time management was viewed as such an important part of the game that we wanted a large graphical representation of the time left

each turn. This was done by adding a large clock in the bottom left corner of the screen. We also made it easier for the players to track their progress by adding four bars that track the four main player statistics. The four bars have icons related to them that light up when the player has achieved a sufficient amount within the certain stat. This was done to give additional indications of progress to the player.

All of these changes were done based on the feedback and our experience with the initial testing we did. These changes were vital to ensure that we got the results we wanted from the main test round.

8.3 The main test - testing the motivational benefits and the educational potential

Our goal for this portion of the testing is to find the answers to the questions that arise in the introduction. We want to know whether educational computer games can motivate learners into using more time on their education. Even though we will not do tests what the students learn from playing our game, it is important that we evaluate what the students feel about the learning potential in our game. For there has to be knowledge to be gained from playing, lest the additional time spent be a waste. Our hope is that our educational game will prove that using educational computer games as a part of the curriculum in courses at a university level is beneficial for students learning process. It is important for us that we can get feedback of the entertainment value of the game, and if they felt that they had learned anything from playing the game. We also want to gather some information on what they feel about the potential for educational computer games in general.

8.3.1 Evaluation technique

In this round of testing we needed to find a way to run the test on a large user group without spending too much time on the actual testing ourselves. We therefore chose to make an Internet site where the testers could download our game, and do an Internet survey. We then used comparison evaluation, comparing the answers up against each other to find out if our game was a success or failure.

8.3.2 Comparison evaluation

To analyze our findings in the survey, we gathered all the feedback and compared the individual comments to look for tendencies in the data. Our results are based on the general

assumptions we can derive from this comparison rather than the individual comments themselves. Some of the questions were in the form of quantifiable data. We used this data to make diagrams and draw conclusions based on their appearance.

8.3.3 How the test was conducted

The number of testers in this test was too many to do the testing on an individual basis. Therefore we couldn't use think aloud technique this time. Our original plan was to visit the students during their group sessions to present our project and observe them as they tested our game. This was no longer possible. Just as the curriculum of the course had changed, the group sessions had also changed, and the students no longer had access to computers during these sessions. Instead we made an Internet site for the test.This site can be viewed in appendix C. At this site we made some instructions on how to download and start the game. Along with this explanation we added some information about the purpose of the test and what the testers where going to focus on while testing. Then we asked the course teacher of INF1050 to add a link and a text that encouraged the students to do our survey, from the INF1050 course Internet page. In addition to this we sent out mail on the course mailing list with a link to the same survey. The actual survey we put up on these pages were a PHP page with questions related to the educational value of our game and games in general. The answers from the survey were sent directly to our email when the students had finished the survey. The questions we asked can be viewed in appendix C.

Test subjects

In this round of testing we did the test on INF1050 students. INF1050 is a course that is normally taken by students who attend the university at their second semester. We went around in the group classes of INF1050 and asked what courses they had taken their first semester, and the most common courses where INF1000 (basic Java programming), MAT1000 (basic mathematics course) and one course more directed against their chosen educational path. As these students are academics the last courses are most often physics, mathematics or informatics.

We do not have access to the exact number of students participating in the course now. There were 160 graduate students from the course in 2006 and we assume that number has not changed drastically since. We had hoped to get feedback from at least 30% of these students, giving us roughly 50 participants in our survey. This was not the case however. It proved to

be difficult to get students to actually answer our survey, and not just play the game. We tried to use different forms of encouragement in order to get more feedback. We presented our project in some of the group sessions. We talked to the group teachers and we sent out emails on their mailing list. In addition to this shad a small prize that we would give to a random person participating in the survey (five FLAX tickets).

This was not enough however and our final number of participants landed at 21 usable responses (Some where discarded because they seemed to have been made in jest. e.g. the content was either incomplete or complete gibberish). This was a lot less than we had hoped for, and less than an ideal number to draw scientific data from. We felt however that we had exhausted our options for reminding or encouraging the students to participate.

Even given this less than ideal number of participants in our study, we still feel we can see some trend in the data, and that those trends function as a basis for our thesis.

8.3.4 Results

General questions on educational computer games

After closing the test round we started compiling all the surveys we got in, and started comparing the data. The first surprising result we saw was that 100% of the testers responded positively to the question: "What do you think about the potential for using educational games at a university level of education?" We expected the census to favour educational games, but this was a land slide. This result can be linked to Prensky's statement that students are ready for educational computer games, but educators are not. It is not possible for us to conclude whether educators are ready for educational computer games or not based on this survey, but the students do seem ready. The answer here can also reflect the fact that answering the survey was voluntary and that students that are interested in educational computer games are more likely to answer a survey on the subject.

Some of the students expressed some concerns as to the difficulty of implementing an educational computer game for their level of education. This was something we expected based on our research. Students haven't seen a lot of educational computer games, and those they have seen have been directed at elementary school students.

This brings us to the next surprising result of our survey. We asked the students: "Have you had any previous experience with educational computer games, and did you learn anything

from playing them?" Almost 60% had never or only once played an educational computer game. In addition to this, none of the students had played an educational computer game based on anything higher than high school level curriculum. Most were based on elementary school knowledge like basic math and language skills.

We can't draw any conclusions based on this survey on why this number is so low. This might be, as Prensky said, the result of educators not being ready to take the necessary steps to include educational computer games into the normal curriculum, but there could be several other reasons. Other reasons might include the failure of designers trying to create entertaining games with educational content. It is evident that educational computer games have failed to reach the masses, as to why this is might be an entire research paper on its own.

The students who had experience with educational computer games were split on their views on the educational benefits of the games they had played. They answers they gave were on either side of the scale, with no students expressing "some" benefits from the games they had played.

> Yes, when I studied french in 10th grade, I didn't learn a thing

> Definitely. Mattegeniet was one of the first games I played and I learned tons of maths from that. The game made math something competitive, and that got me motivated.

It seems evident that there is a huge difference in the quality of educational computer games out there. Third language courses at high school are something that is generally hard to master for students based on personal experience. Given that educational computer games based on teaching language skills were among the first, it is easy to assume that they should be an excellent addition to third language courses. This experience is in contrast to another student who explained that she had learned Norwegian in less than five months because she was motivated by an educational computer game.

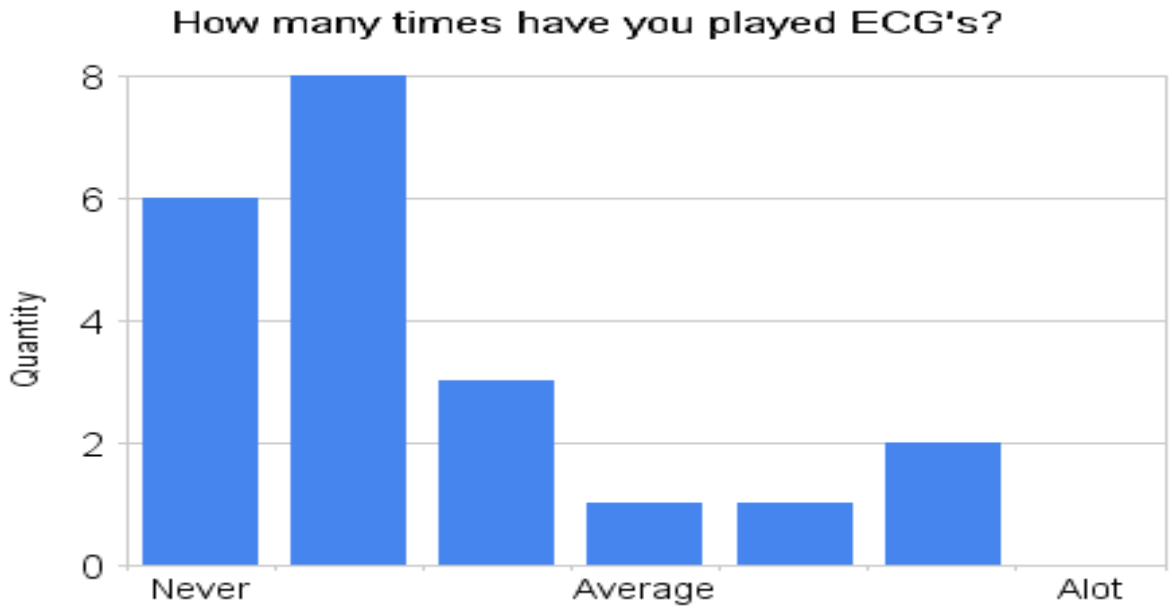


Illustration 27: Test results - Previous experience with educational games

We asked the students to quantify their experience with educational computer games to get some statistics. From this diagram it is evident that students do not have a lot of experience with educational computer games.

Questions related to our game

Viewing the data from the feedback related to our game were interesting. The testers had experiences with our game that ranged from appalling to ecstatic. Reading negative reviews of our game was troublesome, but luckily only two of the respondents where really negative to what we had done. When we compiled all the data, we saw a positive trend towards our project. Looking first at the quantifiable data taken from questions where the user were asked to rate their answer on a scale of 1 to 7. These labels on the charts correspond to the values of these numbers with 4 being average.

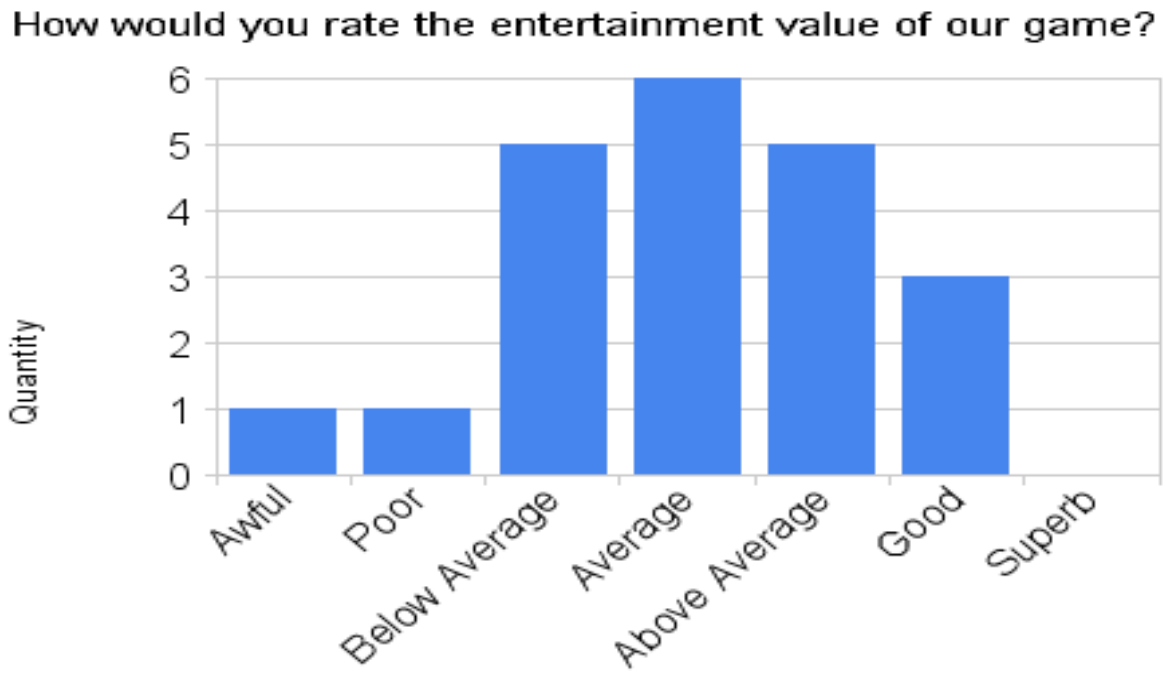


Illustration 28: Test results - Entertainment value

As we can see from this chart, 66% percent of the users rated the entertainment value our game Average or above. This is something we are extremely happy with given that the project had so little development time. With the gamers being used to games with budgets ranging in the millions, we were very pleased that we were able to entertain with our simple game.

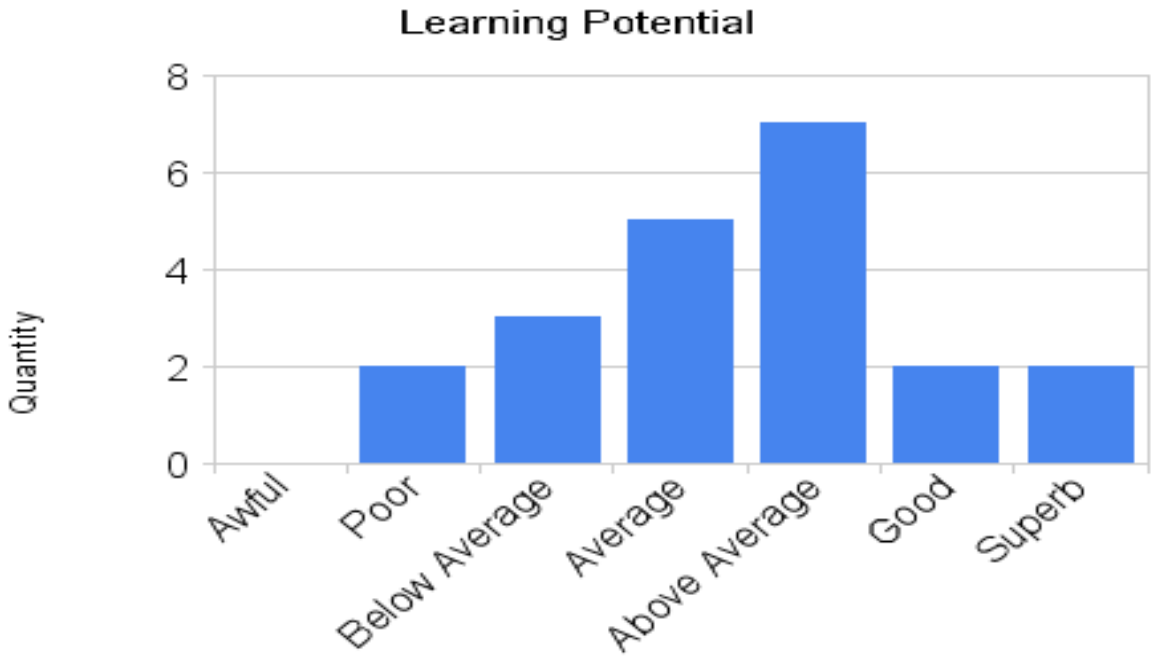


Illustration 29: Test results - Learning Potential

As stated earlier all the students were positive to educational computer games. This is also

something that is reflected in how they view the learning potential for our INF1050 related assignments. We asked the students "How would you rate the learning potential in the assignments related to INF1050?" We can see the table having a median of above average, and 76% of the students marking the learning potential as average or above. It was satisfying for us to see two students marking our assignments with a top score.

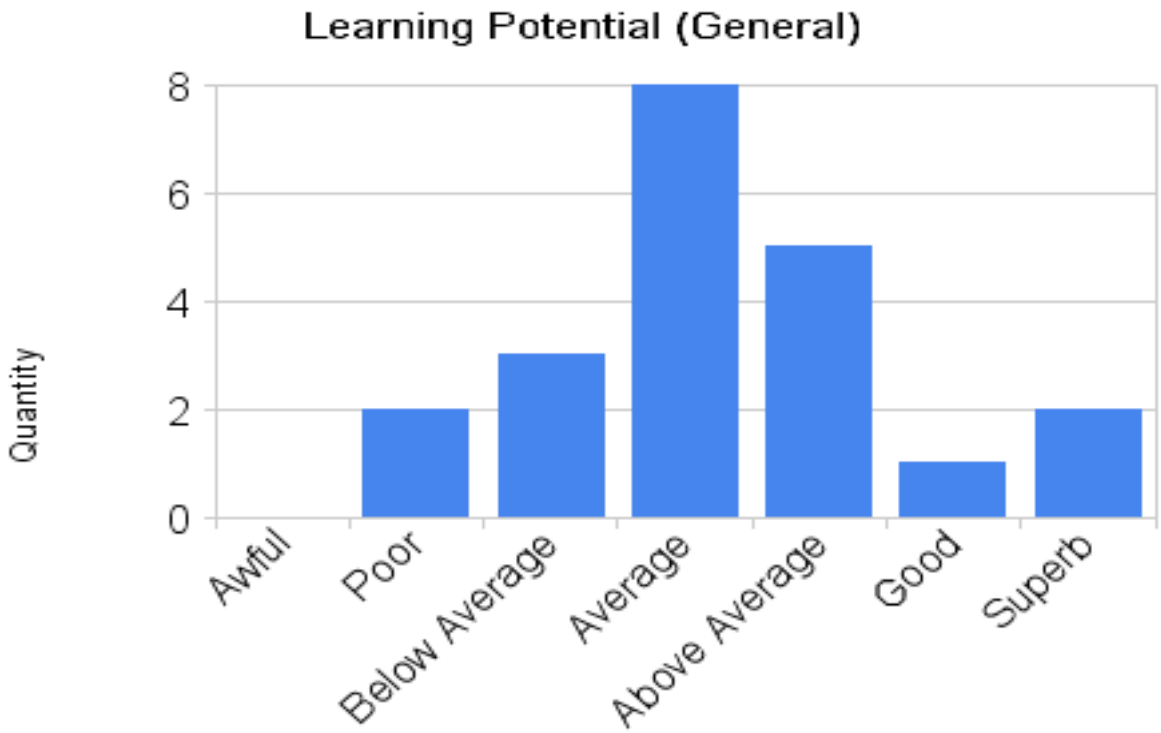


Illustration 30: Test results - General learning potential

Further we asked the students "How would you rate the learning potential for the game on a general basis?" We are a bit uncertain as to why this question got somewhat less optimistic answers than the one regarding our assignments. We half way expected the students to be more positive to questions made by the group teachers or similar.

Looking at the written response to the text questions we saw that they generally reflected what we saw in the statistical results shown above. The first question we asked about our game was: "Would you play this game if it was complete with INF1050 related assignments made by the group teachers? Please elaborate on why or why not."

Only three students responded that they probably would not play the game in that context. This is something we again view as very positive for educational computer games as a whole. Most of the students questioned said they definitely would play the game with assignments made from the group teachers.

> Yes, it's a bit more relaxed than just reading and doing the normal boring tasks. It could even do great as a pause from the normal studying.

Others pointed out certain criteria that had to be met in order for them to play the game as a part of their course.

> Depends on how complicated the quizes or other relations are. Too much thinking takes the fun away.

This answer reflects the findings from the research we did on previous work. If the games are too evolved with learning and forget the motivational aspects, players will get bored and perhaps stop playing completely.

Regarding this question it is also interesting to look at how much time the students used on the correct implementation of the game. This was something that surprised us a lot. Even though a good portion of the testers only used from half an hour to an hour, there were surprisingly many students that had used a lot of time on the game. Some had even achieved high scores that were beyond those of the developers. The notion that the students themselves were positive to playing the game as a part of their studies coupled with the fact that a lot of the students had played the game for such a long time, gives us a clear indication that educational computer games can motivate students into using more time on their studies.

Next we asked the students if they had learned any knowledge directly related to INF1050 while playing our game. This was where we got our biggest disappointment came regarding the answers. Only about 30% of the students (depending on how you interpret some of the answers) said they actually learned any INF1050 related knowledge while playing the game. The most common reason given as to why they had not learned anything was that they simply knew the answers to them already, or that the assignments were too easy. We think this is largely related to the change in the INF1050 curriculum. The new curriculum still was not finalized as we made the assignments and we had to rely on the slides on the home page for information. This resulted in a lot of the assignments being based on content that the students had already been introduced to.

Even though few of the students said they learned any new knowledge, there were several who commented that playing the game was an excellent way of brushing up on their

knowledge. The knowledge the students had learned had mainly been related to UML models in regard to system development techniques. We believe the main reason for this is that we were only able to implement assignments based on curriculum the students had already had lectures on. This was due to the fact that the curriculum of INF1050 was not finalized when we created the assignments. This could obviously be remedied by having the assignments be created by the group teachers of the course using up to date assignments.

To conclude the survey we asked the students about specific things they liked or disliked about the game and gave them a field where they could write general feedback about anything related to the course:

Were there anything you found particularly difficult to understand or hard to do in the game?

Only a few of the participants commented on things they found difficult. The problems they had were mainly related to external things like problems running the game. We had a few students that tried to run the game on unsupported platforms (MAC), or old non-compatible versions of Java. Through email correspondence with these students we got them to either change platform or update their Java version. In addition to this two students commented that some of the assignments were a bit difficult Most of the answers where short and simply stated that there was not any thing particular they had problems with.

> Nope, the game was easy to understand.

> Not really

> The game was easy to understand

We asked the students: «Were there any elements in the game you liked particularly good? Any things that encouraged you to play more?». We got a lot of positive feedback from this question. We list a few of the replies here:

> It reminded me of the brilliant game "Jones - in the Fast Lane". I belive you have stolen plenty of ideas from there.

> The way it's very simular to the good old game: Jones, in the fast lane! With some new features that gave it a new twist.

> I really liked the workchain, you had to get experince, education, nice clothes etc. I wanted it all!

> The random trivia and impossible quiz were entertaining.

> Well in general I like management games. The fact that there are several ways to goal in this game, is giving it replayability.

We were surprised that some students actually saw the reference to Jones in the Fast Lane. It was pleasing to see that we were not the only ones who had fond memories of that classic. This confirmed to us that we had made the right choice in what we based our implementation on. It confirmed for us that we were able to create a game that was entertaining. We were also encouraged by the fact that our goal scheme seemed to have worked in giving the game some re-playability.

> The fact that I could spend time playing a game with content that is relevant to my final exam pretty much gave me all the encouragement I needed to continue playing.

> The software engineering bits were the best, but it was not enoght of it. More actual learning would encourage me to play more

mainly the three different quizes.

I really liked the quizes at the University :)

The three quizes were entertaining

These answers confirm what we concluded based on Gee's research (Gee, 2005). Learning is a huge part of what makes games entertaining. We can clearly see that the educational parts of our game were highly liked by the students. When we created the game we discussed how much of the game should be evolved around learning. We were uncertain whether too many learning elements in the game would take away some of the fun while playing. In the end, the lack of development time prohibited us from incorporating the learning process into a larger portion of the game.

We see now that we could probably have incorporated learning elements in all parts of the game without removing the motivational aspects. Integrating educational content with basic motivational aspects like goals, score keeping and progress ensures that the learning process is part of the entertainment of the game. The assignments we had in the game did just this. Completing them with as few tries as possible is essential to get a good score and progress in the career ladder. The fact that the assignments were related to the INF1050 course did not retract from the fun factor as long as they were related to the gameplay. It actually increased the entertainment value.

The general feedback field was mostly used by the students to wish us luck with the thesis and give us some ideas of some things that could be implemented to improve the game. Most of these suggestions were things we had actually implemented or things we had thought about but could not implement in our time frame. A few students suggested that we make a assignment creator so the group teachers could easily implement assignments for the game.

The version of the game they tested was compiled without the option to create assignments, so it was not obvious to the testers that this particular function was already implemented. This

was done on purpose as we wanted the testers to focus on the game, and not how it would be maintained in a course setting. Some testers wanted a score board, which we initially intended to add, but simply did not find the time to implement.

> Great game, quite difficult, but it is just challenges you enough. Loved the quizes :)good work

8.3.5 Evaluation

We were pleased with the results of the test we did on the INF1050 students. Even though the feedback we got indicated some mixed feelings towards the game we created, most of the feedback was positive. A lot of students saw the potential in the game, and how it could be used as part of the INF1050 curriculum with assignments created by the group teachers.

The number of testers that answered our survey is too small to create any substantial statistical data, but we feel it is large enough to be used to indicate tendencies. The tendencies we saw in our test were that students are positive to educational computer games as part of their education. Even at a university level.

Our game could definitely use some improvements. We still feel that the test show that our game has potential as an educational tool. Some changes might be necessary to improve the learning potential. The main improvement we can see based on the feedback we got is that the educational content needs to be a bigger part of the game. We got surprisingly positive feedback on the assignments we had implemented, we just needed more and more parts of the game should be based around such assignments.

We did not test what the students actually learned from playing our game, but still some of the testers commented that they did. The test at least shows that some of the students used a lot of time on our game and could brush up on INF1050 related knowledge while having fun. If an educational computer game can be used to encourage students to use more time on their studies, it is almost certain to have a positive effect on their learning

8.4 Summary

In this chapter we went through the two test rounds we did. The first test was performed on two students at our university. The focus on this test was to evaluate the gameplay and user interface. The first test we did helped us prepare for the final test on the INF1050 students. Several changes were made to the game based on the feedback we got from our first test. These changes were a vital part of making the second test round the success it was.

The second test round was done on 21 students enlisted in the INF1050 students. This test were the basis for testing the research questions in this paper. The results from this test was presented in this chapter and will be used as a basis for the discussion in chapter nine.

Chapter 9
Discussion

In this chapter we will discuss if our educational game meet the specifications we described in chapter four. Further we explore if the game is appealing to the players, and find out if the game is suited to use as an educational tool. We round up the chapter by looking at what we could have done differently in the development process of the game.

9.1 Does the software meet the specifications?

9.1.1 The system design characteristics

User friendly

What we learned from the preliminary testing is that the game as it was then was not user friendly enough. It lacked information about what the player was supposed to do and how he/she did it. We did a major remake of central themes concerning information pop-up boxes, and we also made an extensive tutorial on how to play the game.

We also made a guide on how to download and get the game started, but excluded information about how to install Java. As this game is developed for people attending universities we chose to assume that the persons already got basic understanding about computers and know how to use them. If we were to include a lot of information about topics the players already know, they could potentially get bored of all the excessive information and not read what we really want them to read.

The test we did on the larger user group showed us that the students understood easily what to do in the final version of the game. Most of the feedback we got from the students was related to the actual game, such as:

> The game was great, even though it seems like it was a tough neighbourhood, damn muggers!

Few people experienced any problems with actually understanding how to play or any problems with getting the game started. There are still some things in the game the students were uncertain about, but they actually liked the fact that everything wasn't explained so they had to find out for themselves.

A problem that two students encountered was that the game is not MAC compatible. We were quite aware that we had not tested it on a MAC computer, due to other more important tasks we chose to ignore this. We see that if this is a game that would be used as an educational tool at a learning institution it needs to support all OS-platforms. Nothing is less user friendly than a game that will not run because you use a different OS than most others. On a side note here, we were aware that most computers at campus on UiO are run on Linux and Windows, therefore we chose to spend our time to make the game as user friendly as possible on those operative systems.

Extendable

Before releasing the final version of the game we read through all the code we have made and added comments to everything. As the code is now, we believe that anyone who know basic Java could read and understand our source code. Along with the code comments, we have added extensive explanation of the game code in this thesis. With the class diagrams and explanation of what the different classes represent in the game it should not be a problem to extend the code to use in further works.

Another fact that makes the game easy to extend is that all the places and course names in the game are given in text files. The documentation on how the text files work is also included so even a person that does not have prior knowledge to Java can go in and edit our game content. This makes the game easy to adapt to teach other knowledge than just System Development as we chose to do.

Reusable

A goal for us was that our game could be used in later project as well as for our purpose of testing the value of educational games on a university level. We therefore made sure that the educational content in our game could be switched out with other content.

Our solution here was to put the actual educational question content into text files. This means that the game content can be changes by just manipulating the text files and adding images. To ease the process of editing the content we made the assignment creator. The assignment creator allows persons with administration rights to edit and add new questions to the game. This makes sure that the content of the game can be in constant evolution, and make sure that it is well suited for reuse. Potentially a university course could use this game as a standard game and assign group teachers to update the game with course related question and content.

We also thought of the possibility of adding functionality so the game updated itself through a database. But for our testing purpose we found out that if people want to download the game and play it from somewhere they do not have Internet connection they should be allowed to do so.

A drawback to our game is that the version we released does not have a save functionality. If questions is just added to the old content and the player needs to re-play the old content before advancing to the new content it could be very repetitive and boring. A save function shouldn't be very hard to implement. All information about how far a player has advanced in the game is stored in two-three central Java classes.

Entertaining

The testing we conducted in the second test round mainly focused on how the player perceived the game. Our main concern was if they thought the game was entertaining and if they liked it as a supplementary to their regular course curriculum. The feedback we got from our test subjects mostly indicated that they enjoyed playing our game:

> Great game, quite difficult, but it is just challenges you enough. Loved the quizes :)

Besides being entertaining, they liked the fact that they could learn course related knowledge in some other way than just reading the book and lecture slides:

> The fact that I could spend time playing a game with content that is relevant to my final exam pretty much gave me all the encouragement I needed to continue playing.

The negative feedback we got was mostly in forms of bugs in the game that prevented the player from advancing in the game. One of the test subjects had problems with advancing past a certain question, and therefore quit the game. This shows how it is important that there has to be good feedback to the in-game questions to keep the attention of the player.

The game itself might not be the most entertaining in the value of nice graphical effects and such, but with the added element of educational content we feel that the player is encouraged enough to continue playing the game.

9.1.2 Functional requirements

The game:

Type of game

In the game you create and take control of a fictional avatar that you lead through the game gaining different education, work and work-experience. In order to advance in the game you need to answer questions that teach different knowledge depending on what the game administrators have implemented to the game. The main educational aspect in the game is located at the in-game university, and through this the goal of the game to make the player learn educational knowledge is reached.

User interface

The user interface of the game is a graphical window user interface developed in Java. The user interface gives the player information about where he/she is located in the game, where he/she can move and what actions that can be taken at the different places. The UI also gives information about the current state of the player, such as how much money the player has and far the player has advanced in order to reach the winning conditions in the game. The UI also have a drop down menu where the player can find a thorough game tutorial and do actions like close the game, make a new game and such.

We reached our goal of making the game compatible with both Windows and Linux. As mentioned earlier MAC users experienced some problems, but at this was not a central goal of our project we still believe we reached our goals with the user interface.

User input

The player is able to use to computer mouse and keyboard to interact with the game.

Answer educational questions

The game is able to give educational questions and provide feedback. The game version we tested on the student should probably have had better feedback than just if they got the answer right or wrong.

Scoring and goals

While the player is playing the game he/she can at all times view how far from reaching the goals he/she is. This can be viewed through the lever bars on the left hand side of the main user interface. When the player quit or completes the game he/she is displayed with a summary on how well he/she played and given a final score. This score is based upon how far the player has advanced in the game and several other small factors such as if he/she bought the best house in the game.

The assignment creator:

Due to lack of time we chose not to use too much time on the assignment creator and rather focused on the game itself as we thought that was more important for our project.

The purpose

The assignment creator is able to add questions and exams into the game. Due to time limitation we did not add a feature that makes it possible to edit already existing questions. Because of this one manually have to go into the text files where the questions are stored in order to change the content of the game.

Availability

The assignment creator is only available to administrators of the game. As it is now one need to recompile the game in order to change the admin flag, and get access to the assignment creator.

User interface

The administrator is able to add questions to the game through a window based GUI. It is also possible to choose where the questions should be added in the game.

Administrating the questions

A person with administrator rights is able to add questions to the game, and chose where they should be added. We did not have time to implement a feature that lets the administrator view the questions that are already in the game. There is no way to change where in the game a

certain question shall appear to the player without editing the text files manually. There is neither a way to edit or delete a question through the assignment creator.

Add images

The assignment creator is able to add images to a question through a GUI interface and file chooser. The assignment creator also got a GUI to aid the person creating the assignments to find points of interests in the images added.

9.2 Is the game appealing to the players?

As we found out in chapter two, we needed the game to be motivational in order for it to be appealing to the players. The key aspects we found were challenge, goal, feedback, curiosity and uncertainty. We believe that all these aspects are more or less present in the game we developed. The players are challenged through the questions in the game, and indirectly through the scoring system and their personal desire to do their best. These challenges were the major fun factor for most of the testers of our game:

> I liked the quizzes at the 'University' pretty good :)

The goals we implemented in the game also added to the game appeal amongst the testers. Some of the testers even played the game longer and got a higher score than we developers did. As we look at the comments from the testers, we see that several complained that they had to spend too much time on gaining money and eating food. This ruined some of the fun for some of the players. This issue lies in the balancing of money, education and work within the game. From what we see now after the testing is done, we could probably have made the game much easier when it comes to obtaining money, and rather put more effort into making more educational questions. After all, the most entertaining and most encouraging for the gameplay were the challenges that went directly on the players knowledge of system development:

> I probably would have played the game if it was more focused around the system development assignments, and not work/food/rest.

This show how the students' appeal towards educational computer games also comes through the potential gain of knowledge.

Another important aspect of educational games is feedback. As mentioned earlier, our questions did not have any other feedback then feedback on whether the player got the question right of wrong. As a result, some of the testers felt that they lacked the proper feedback. When for instance when faced with a multiple choice question, one of the students just redid the question until he got the answer right. He explained how he did not learn anything while doing so. The question just became an obstacle he advanced past by answering the question at random without thinking about why or what he actually answered:

> No, as the parts I hadn't learned yet through the course were easily skipped during my quick play-through by playing "Master Mind" with answers.

As a whole we feel that we failed a bit on giving proper feedback in our game, but we got the verification that good feedback is an important factor when it comes to educational computer games. The feedback we gave the player was in the form of "You answered the question correctly." or "You answered the question wrong." To improve on this we could have been better at giving text feedback if the player answered wrong. Another nice improvement that could be added is to make it possible to attach a solution-image to the questions.

We were satisfied with the way we implemented curiosity into the game. The testers told us that they really enjoyed the questions, and were encouraged to play more just to see what questions that came next. Further we noticed that several of the testers liked the fact that you had to switch jobs all the time, and that it was difficult to know which jobs to apply for. The fact that you have to apply to the new jobs and that the criteria for getting the job were partly based on numbers hidden to the player, resulted in anticipation and curiosity for the player. This illustrates how our findings are in line with Malone's *Optimal level of information complexity* (Malone, 1980) and Gee's *Information just in* time. (Gee, 2005) Curiosity and anticipation are important factors when making educational games.

When it comes to uncertainty we feel that the game is mediocre. The main uncertainty factor in our game is that it has a very high number of move options. This is also making the game more re-playable. The players can learn from their previous errors and replay the game with

another tactic. There are many ways the player can reach the different goals in the game. Since we made the game quite easy, we wanted the testers to advance far into the game, there were little uncertainty if the main goals were reached or not. The only thing that prevented the player from finishing the game was if you had too little money, or was unable to advance past a question. We added some uncertainty to the game by adding the chance of getting mugged or that the player's house was being robbed. This happened a little too often, and resulted in some testers complaining about it. Some of the testers also complained that they got the same questions over and over again if you failed. As the game is now it is possible to add several questions to the same assignment level, and make the question appear random from the given assignment level. As we have this only a few places in the game which the testers tested, few noticed it. This shows how it is important with uncertainty and make sure that the player is not bored with repetitive content while playing.

Overall we were quite satisfied that the testers in general liked the appeal of the game. We also noticed how the educational aspect of the game was the main motivating factor for the testers:

> I would definitely have played this game if the group teachers updated it with content that is relevant to what we're doing right now. [...]
> It would make it easier for me, as I have a job on the side and it'd motivate me more if I could just launch the game and have that weeks
> questions loaded in and be displayed as practical and theoretical tasks (accompanied with the answers though)

9.3 Does the game work as an educational tool?

Our goal with this game was as mentioned earlier not to test the educational value of our game, but rather find out if the use of such educational games can stimulate the students to spend more time on the course and through this learn more. Due to the fact that we did not see the complete curriculum of INF1050 before our game was complete, we were reluctant to add assignments based on curriculum we were not sure were going to be a part of the course. This made us add content that the students already were introduced to in the course lectures, before they played our game.

One of the main benefits of using an educational computer game is most probably the fact that the player gets almost instant feedback on their work. If the player is given good enough feedback to reflect upon, the learning potential is great. When it comes to our game it lacks some of the extensive good feedback to become a good tool to be used as an educational game to teach new knowledge. On the other hand we experienced that several of our testers liked the game, and used it as a tool to repeat knowledge they had already learned from the course lectures. When asked: "Did you learn any INF1050 related knowledge while testing our game? If you did, what did you learn?" two of the testers answered:

> I had to check my book some times to be sure that my answers was correct. So it worked like a revision of the syllabus

> I didn't play the game for that long, due to the fact that it's easter and all ;) I did however get to brush up on what I already knew and learn
> some good uses for the different development models that will be handy for the final exam :>

There were several testers that indicated the same thing as the two above. This shows how the game could be used as spare time leisure to improve the time spent on course related knowledge. Few said that they learned anything new. This could come from the fact that the testers were presented with topics that they had already learned, or the fact that the feedback in the game was lacking in quality. Another important thing to mention here is that we chose not to implement the advanced question types. If the question types had encouraged more advanced thinking the students might have learned more. At the same time we are in the belief that too advanced questions could have a negative effect on the players also. They liked the fact that the game was easy enough to complete without constantly referring to lecture material in order to answer the questions correctly:

> Too much thinking takes the fun away.

We believe that the game we developed proved to be a tool for repeating already known knowledge. That said we also believe that the value of such games should not be

underestimated. One of the goals of this project was to see if an educational game made the students spend more time on the course. Several of the students liked the idea and said that if the game was extended with more course related questions, they would play it.

If a large educational game company used their resources and at the same time focused on the elements we have found important for educational games, we believe that educational game that they produce could be more successful. Such companies have much larger work capacity, money and knowledge on how to make games, they could also stimulate the players even more with advanced graphics and other such elements. This would make the non-educational parts of the game more entertaining.

9.4 What could have been done differently?

A weakness of this point is that we had relatively few testers in the final round of testing. Only 21 testers for such a game are a little few to get good presentable statistics. It was very difficult to get the students of the INF1050 course to take the time to test the game. If we could have done it differently, we would have focused on getting the game finished sooner, and given the testers more time to complete the test. Still we feel that we got enough test results to get an indication on how useful an educational game at a university level could be.

We believe that we could have used more time on balancing money, work and education. This way we could have gotten more constructive feedback from the testers. Some of the testers simply gave up due to the fact that they kept getting stuck when dieing due to lack of money. As a solution to this we would probably have added a way of setting the difficulty level of the game. This would make the game easier for those who did not manage the level we set for the game, but at the same time not bore the players who thought the game was good as it was.

Another important thing we would have used more time on was to create more questions with better feedback. This would potential give us more information on how well the game was suited to not only repeat already known knowledge, but also teach new knowledge. We found out that educational games is good for making the students spend more time on their studies but with the additional feedback we could have learned more about the learning potential.

Now as the development of the game is finished, we still have a lot of ideas on how to improve the actual game experience. This includes more random events, more ways to evolve the avatar, better graphical user interface and more rewarding elements in the game to

encourage gameplay to mention a few. But, as we have had limited time on developing the game, we see that we had to chose what we thought was most important, and are generally satisfied with what we did.

Chapter 10
Conclusion

In the later years computer games have becoming a larger part of our society. We can see this from the direct financial contribution of the gaming industry, and from how computer games are becoming more and more a social acceptable venue. The common view of computer games is that it is used just for leisure and fun, but good computer games can also be learning machines which challenge the player to learn new knowledge. This unique feature of computer games is something that educators can not afford to leave untested.

As we researched earlier work within the field of educational computer games, we found that most of the research that had been done on students at a high school level or a theoretical level. When one look at Ladd's work (Ladd, 2006) on using game related subjects as a part of a university course, one can clearly see that it is a huge potential in adapting educational computer games to a university level as well. Through our own research on educational computer games we decided to test the potential of educational computer games at a university level.

As we tested the game on the students in INF1050, we clearly saw a tendency of how good educational computer games can be. Despite the fact that only a relatively small amount of the students had prior experiences with such games, our test results showed that those students who responded are positively inclined towards educational computer games. Further we also found out that if an educational game is to be used regularly as an addition to regular tutoring, the game needs to be really good. With good, we mean that it needs to motivate the students, be entertaining, and at the same time teach the students new knowledge or brush up on old.

This brings us to the conclusion of this thesis. Did we find an answer to the research questions we presented in chapter one?

1. Is it possible to make an educational computer game for a specific course at a university which increases the students' motivation towards learning and the time they spend on studies? Will this game contribute to improve their learning?

We believe that educational computer games can be used on students on a university level to increase their motivation towards learning, if the game is good enough. We do not believe that educational computer games can replace our standard teaching methods, but it can be used as a tool to motivate the students to spend more time reading up on the school subjects. Most of the students we encountered during our test showed a desire to perform well in their courses. Even though our game was lacking in the tutoring of new knowledge, it motivated the students to use their spare time to brush up on old knowledge. We clearly saw through our testing, that there is a huge potential for using educational computer games to increase the time students spend on their studies. It is a known fact that students who spend more time on subject also becomes better at a subject. We believe that educational games can improve their learning because if the students spend more time on a subject they also become better at it.

2. Can the knowledge we learn from our project say anything about educational computer games in general? If so, what does educational game designers need to focus on in the future, and how can this be used to make educational game for students at a university level?

Through the development of this game and our testing, we see that the most important factors in an educational computer game are entertainment and motivation. The entertainment needs to be present in order to keep the attention of the player, while motivation encourages the player to play the game. There are very many game elements that provide entertainment and motivation. Primarily entertainment is provided through exciting and rewarding gameplay that stimulates the players sense of achievement. These findings are also backed up by what we read in Malone (", 1980) and Prenzky's research (", 2005). Motivation can come in many forms, but in an educational computer game the main motivational drive often prove to be the learning of new knowledge. We believe that if one merge the aspects of motivation, entertainment and education in a good way, it is possible to make educational games that students at a university level would enjoy and learn knowledge from while playing.

10.1 Future work

Here we will look at the future for our game and how our project can aid future research on the use of educational games at university level. We also point out some aspects of the game that we believe can improve the educational game further

Important improvements to the game

Here we will look at important improvements to the game that we believe would motivate the player more and thus improve the educational value. All of these improvements were not done by us because we had a limited time developing the actual game.

GUI

The graphical user interface of the game as it is now is pretty basic. To add more fun into the game it could be possible to extend the GUI with more graphical elements. For instance make animation to the action of moving around in the game. More graphical elements and animation will make the game feel more interactive and thus stimulate the player more.

Sound

As the game is now there are no sounds except standard sounds for the window-frames in the operating system. We believe that adding sounds to different actions and maybe some background music could make the game more enjoyable to play. At the same time we know from experience from playing games ourselves that sound can also be an annoying factor, so the game-sounds should be possible to mute.

Balancing and difficulty levels

The balancing of the different elements in the game such as money, food-cost and similar values are not optimal in the game as it is now. This is a rather difficult job to do, and can totally change the feel of the game. To improve the game experience for the player this could be looked more into. To ease this job we have put all important balancing attributes at the same place in the code.

Further we believe that the game should support different levels of difficulty. This could be added at the start of the game, for instance force the player to choose between easy, medium or hard. This could then be reflected in the amount of money one get from the different jobs, or the difficulty level on the educational questions in the game. As the game is now there are no way of changing how hard the game is to play, and if some players find the content of the game to hard a challenge they might get frustrated and quit the game.

Operating system support

If the game we have developed should be used in an educational setting it is important that it is supported by the most common operating systems. All the potential students need to be able to run the application on their home computer. We experienced from our testing that the students used several different operating systems, and some complained that the game did not run on their personal computer. The current release of the game does not run on Mac. Since a lot of students own Mac computers it is essential that the game is changed to also support Mac before it is used in a tutoring situation where the students are expected to play the game on their personal computer.

Several question types

To further evolve our game we have made it easy for later developers to add new question types to the game. New question types make the game-play more varied, and thus more entertaining to play. This addition also opens up new possibilities for testing different forms of knowledge. For instance one could add a question type that asks the student to write longer texts, parse and correct them accordingly to the answer. Another type of question that would fit nicely into our game is questions that let the user drag and drop elements on the screen.

Better feedback on questions

The feedback in our game could be improved on a lot. As mentioned earlier, a solution to this is to add textual feedback to the questions, or the possibility to attach solution images to the questions.

Use in practice

We have tested and concluded that the game itself is a nice educational tool for motivating the students more in their work. A natural extension of our work would be to integrate our game in university education. For instance have a group teacher constantly update the game with new content, and use it as an obligatory element to a university course. This is a field that is interesting and may potentially prove to be a new angle to the teaching of university students. A nice feature that could be added here is a database for the questions. If the game downloaded the questions through an online database it would be easier for the administrators of the game to update the questions and thus make the game more evolving.

Bibliography

«Bygfoot», 2007, «*Bygfoot : A Football Management Game for Linux*». <http://bygfoot.sourceforge.net/>

«Colobot», 2006, «Colobot», Epsitec games. Last visited May 2008. <http://www.ceebot.com/colobot/index-e.php>

Djupedal, Ø. 2007, «*A proactive budget for Norway's knowledge economy*». Ministry of education and research, Press release No.: 68-07. Available at: <http://www.regjeringen.no/en/dep/kd/press-contacts/Press-releases/2007/--A-proactive-budget-for-Norways-knowled.html?id=482558>

«Eclipse», 2008, «*Eclipse – An open development platform*», The Eclipse Foundation. Last visited May 2008. <www.eclipse.org>

Entertainment Software Association, 2008a. «*Entertainment Softwawre Association*». Last visited May 2008. <http://www.theesa.com/about/index.php>

Entertainment Software Association, 2008b, «*2007 Essential Facts About the Computer and Video Game Industry*» Entertainment Software Association. Available at: <http://www.theesa.com/archives/ESA-EF%202007%20F.pdf>

«Football Manager», 2007, «*Football Manager2008 – Game info*». Last visited May 2008. <http://fm08.footballmanager.net/en/article/101/1583.html>

«Fronter», *2008, «Fronter Open Learning Platform»,* Fronter AS. Last visited May 2008. <http://www.fronter.no/no/index.html?m!http://fronter.info/no//products_start/Konseptet.html$l!products/menu.html$top!products>

«GameEditor», 2008, «Game Editor – Cross Platform Game Creator» Game Editor. Last visited May 2008. <http://game-editor.com/>

Games2train.com, 2008, *«Games2train.com».* Last visited May 2008. <http://www.games2train.com/site/default.html>

Gee, J.P. 2005, «*Learning by Design: good video games as learning machines*». E-Learning, volume 2, number 1, 2005. Available at: <http://www.wwwords.co.uk/pdf/viewpdf.asp?j=elea&vol=2&issue=1 &year=2005&article=2_gee_elea_2_1_web&id=84.202.13.192>

«GoogleDocs», 2008, «*Google Documents*», Google. Last visited May 2008. <http://docs.google.com/>

Gurholdt, G. And T.E. Hasle, 2003, «*Grunnleggende systemutvikling*». Cappelen, ISBN: 82-02-19868-2. Chapter 1-11, chapter 14-19, subject 1-4 and subject 7-9.

«INF1050», 2008, «*INF1050 - Systems Development*», Department of Informatics, University of Oslo. Last visited May 2008. <http://www.uio.no/studier/emner/matnat/ifi/INF1050/index-eng.xml>

«INF1050 exams», 2007, «*INF1050 – Tidligere eksamensoppgaver*», Depertment of Informatics, University of Oslo. Last visited May 2008. <http://www.uio.no/studier/emner/matnat/ifi/INF1050/tidligere_eksamensoppgaver /index.xml>

«JAR», 2008, «*Packaging Programs in JAR Files*», Sun Microsystems. Last visited May 2008. <http://java.sun.com/docs/books/tutorial/deployment/jar/>

«Java applet», n.d., «*Applets*», Sun Microsystems. Last visited May 2008. <http://java.sun.com/applets/>

«Java Web Start», n.d., «*Java Web Start Technology*», Sun Microsystems. Last visited May 2008. <http://java.sun.com/products/javawebstart/>

«Jones in the Fast Lane», 2002, «*Jones in the Fast Lane*», Vintage-Sierra.com. Last visited May 2008. <http://www.vintage-sierra.com/other/jones.php>

Korzeniowski, P. 2007, «*Educational Video Games: Coming to a Classroom Near You?*» TechNewsWorld, available at: <http://www.technewsworld.com/story/56516.html>

«Kramer», 2008, «*Wikipedia, the free encyclopedia -Wolfgang Kramer».* Last visited May 2008. <http://en.wikipedia.org/wiki/Wolfgang_Kramer>

Kramer, W., 2000, «*What is a game?»*, The games journal, translated to English by J. Tummelson. Available at: <http://www.thegamesjournal.com/articles/WhatIsaGame.shtml>

Ladd, B.C., 2006, «*The curse of Monkey Island: holding the attention of students weaned on computer games*», Journal of Computing Sciences in Colleges, v.21 n.6 p. 162-174. Available at: <http://portal.acm.org/ft_gateway.cfm?id=1127464>

Landro, A., 2007, «*A game generator – the framework for an educational system development game*», Department of Informatics, University of Oslo.

Løwe, T. and J.P. Sæther 2007, «*Studenters inntekt, økonomi og boforhold. Studenters levekår 2005*» Statistics Norway, available at: <http://www.ssb.no/emner/00/02/rapp_200702/rapp_200702.pdf >

Maciaszek, L. A., 2007, «*Requirements Analysis and Sysem Design*», Addison Wesley, ISBN: 978-0-321-44036-5. Chapter 1-4 and chapter 7-9.

Malone, T.W., 1980, *«What makes things fun to learn? heuristics for designing instructional computer games*», Symposium on Small Systems Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, p. 162-169. Available at: <http://doi.acm.org/10.1145/800088.802839 >

«MVC», 2002, «*Java BluePrints Model-View-Controller*» Sun Microsystems. Last visited May 2008. <http://java.sun.com/blueprints/patterns/MVC-detailed.html>

Nobelprize.org, 2008. «*Nobelprize.org - Educational Outreach* Program». Last visited May 2008.

<http://nobelprize.org/nobelweb/edu_program.html>

«PHP», 2008, «*PHP : Hypertext Preprocessor*», The PHP Group. Last visited May 2008. <www.php.net>

Prenzky, M., 2002, «*Marc Prenzky.com*», Marc Prenzky.com. Last visited May 2008. <http://www.marcprensky.com/>

Prenzky, M., 2005, «*In Educational Games, Complexity Matters. Mini-games are Trivial - but "Complex" Games Are Not. An important Way for Teachers, Parents and Others to Look At Educational Computer and Video Games*», Educational Technology, vol. 45, no. 4, 2005. Available at: <http://www.marcprensky.com/writing/Prensky-Complexity_Matters.pdf>

«Regex», 2004, «*Regular expressions*», The IEEE and The Open Group, The Open Group Base SpecificationsIssue 6. Available at: <http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html>

Skagestein, G., 2005, «*Systemutvikling – fra kjernen og ut, fra skallet og inn*», Høyskoleforlaget, 2nd edition, ISBN: 82-7634-671-5.

Stenersen, R. And K.S. Pedersen 2007, «*Ny undersøkelse om studentenes levekår. Mener studentene må jobbe mer*» Verdens Gang, available at: <http://www.vg.no/nyheter/innenriks/artikkel.php?artid=178572>

«Sun», 2008, «*Sun Microsystems*», Sun Microsystems. Last visited May 2008. <http://www.sun.com/>

«SVN», 2006, «*Subversion*», CollabNet Inc. Last visited May 2008. <subversion.tigris.org/>

The Norwegian Association of Students, 2007, «*Studentenes levekår*». The Norwegian Association of Students. Last visited May 2008. <http://www.stlweb.no/article.asp?w=8601839>

Tuzun, H. 2003, «*Motivating learners in educational computer games»,* Annual Proceedings of Selected Research and Development Papers, Presented at the National Convention of the Association for Educational Communications and Technology, p. 465-475. Available at: <http://yunus.hacettepe.edu.tr/~htuzun/html/academic/Motivation-AECT2003.pdf>

Vasilyeva, E, 2007, «*Towards personalized feedback in educational computer games for children*», Department of Computer Science and Information Systems, University of Jyväskylä. Available at: <http://delivery.acm.org/10.1145/1330000/1323264/p597-vasilyeva.pdf?key1=1323264&key2=7093206021&coll=&dl=&CFID=15151515&CFTOKEN=6184618>

«W3C», 2008, «*World Wide Web Consortium – Leading the Web to Its Full Potential...*», The World Wide Web Consortium. Last visited May 2008. <http://www.w3.org/>

World of Warcraft, 2008, «*World of Warcraft Guide*». Blizzard Entertainment. Last visited May 2008. <http://www.worldofwarcraft.com/info/basics/guide.html>

«XML», 2008, «*Extensible Markup Language - XML*», The World Wide Web Consortium. Last visited May 2008. <http://www.w3.org/XML/>

«ZQL», 1997, «*Zql: a Java SQL parser*», J. Hoffman. Last visited May 2008. <http://www.experlog.com/gibello/zql/>

Appendix A
The tutorial

index.html:

Tutorial

Index

- Index - This page
- Overview - A brief explanation of the elements in the game
- Main window - Explain the different elements of the main window and what one can do
- Places in the game - Overview of the different places in the game, and what one can do there
- Housing and rent - Explain how housing and rent works in the game
- Work - Explain how to apply for a job, and how to work
- Food - Explain the different aspects of food and eating in the game
- Furniture - Explain how furniture works and why you need it
- Education - Explain how to go forth to get an education
- Assignments - Explain how the different assignments work
- Bank - Explain how the bank works and benefits one gain
- Clothing - Explain how clothing works in the game
- About the game - Info about the game, authors and why it was made

---

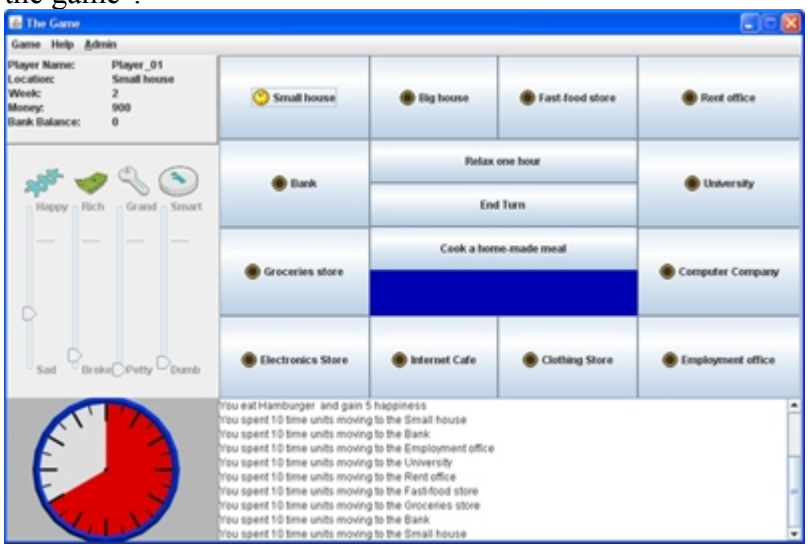
Continue on to Overview ---->

overview.html:

Overview

Overview

In this game you take control of the life to a person trying to obtain a good job and education. You need to guide this person to a better education in order to get better jobs. At the same time you need to work in order to pay for your house and buy food. Along the way you will be asked questions you need to answer correctly in order to advance in the game. The main elements of the game consist of 4 different goals you need to get points in, in order to "beat the game".



The different goals in the game

- Work experience

You obtain points in work experience by working. Different work positions give you different level of work experience, as a rule of thumb, jobs that have better wages give better work experience. When you get a job, you need to work there for a pre-defined (hidden from the player) number of times before you obtain the work experience points for that work position. When applying for a work position you often need work experience, from jobs that pay less, in order to get the job. Your current work experience is given by the job you have had that gave the best work experience.

- Education

You obtain points in education by attending courses at the "University". When paying to take a course one is often given an assignment you need to answer correctly in order to advance. Some of the courses also consist of several different questions. (In some courses you are given a random question from a certain set, from others you need to answer them all

correctly)

 **- Money**
You get points in Money by having money. Your points in Money is directly linked to the size of your wallet.

 **- Happiness**
Happiness is controlled by many different elements hidden throughout the game. One can both gain happiness and loose happiness.

Continue on to Main window ---->

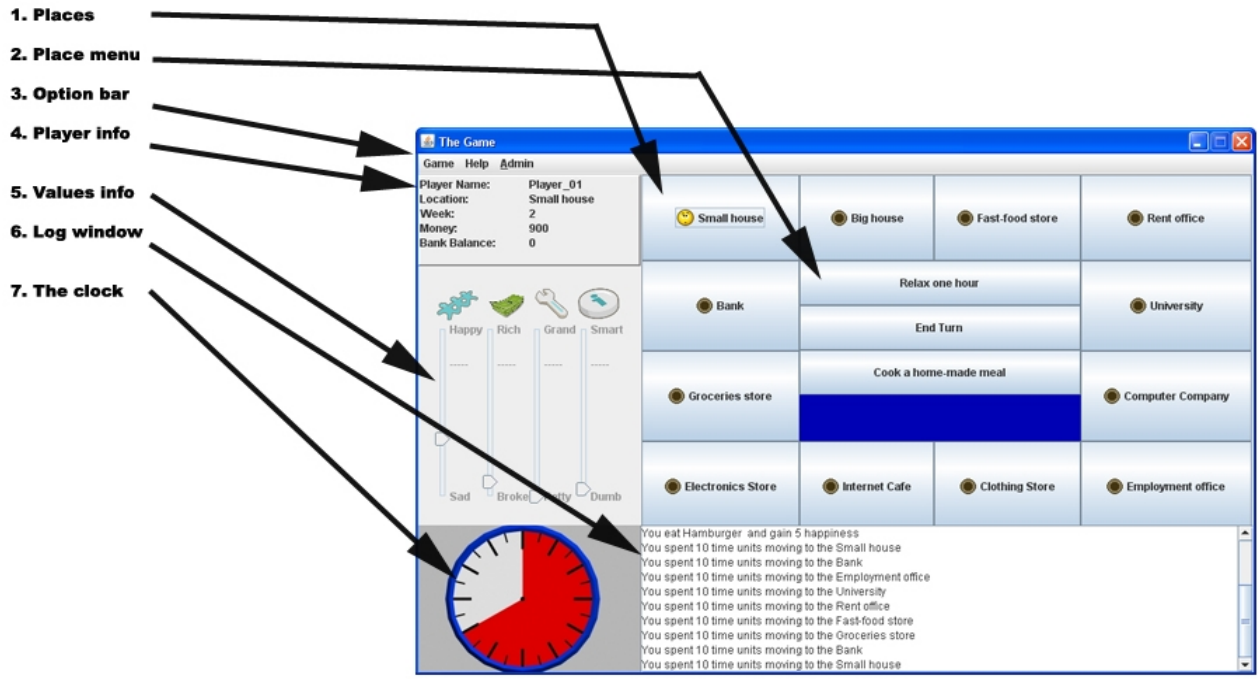
main_window.html:

Main window

Welcome

When first starting the game you need to type in an avatar name. This is the name you choose to give your avatar, and have little influence on the game-play. Then you are given a brief explanation on the background and future goals of your newly created avatar, and you move on to the main game window.

The main window



- **1. Places** These 12 buttons represent the different places you can visit in the game. The smiley is representing your avatar and show where you are located.
- **2. Place menu** The center button row represent actions you can perform at the location your avatar is currently visiting.
- **3. Option bar** Standard drop down menu where you can open different information windows, start a new game and such.
- **4. Player status** Information about your avatar and time.
- **5. Values info** Show the 4 different goals in the game, and a bar that indicates how many points you have in the different goals. When you gain enough points to beat the game in a certain goal the icon will change to a glowing icon. The amount of points needed in the different goals is indicated by the dotted line.
- **6. Log window** This window will log all actions in the game.
- **7. The clock** Show how much time that is remaining of the week. When it is fully red, the week is over, and you are moved home to your homeplace.

Continue on to Places in the game ---->

places.html:

Places in the game

Places in the game

- **Small house** Your starting home place with low security system. This place is often visited by burglars that steal your belongings. If you have a fridge, you can cook meals at home, and save time to do other important stuff.
- **Big house** A bigger, but more expensive, house with an upgraded security system. Burglars are reluctant to break into this house, but some brave souls still try.
- **Fast-food store** A place to buy food.
- **Rent office** The place you go to pay the monthly rent for your house. This is also the place you want to visit if you want to rent a new house.
- **University** The school in the game. You go here to get an education.
- **Computer company** A place that offer good jobs. In this release of the game, this place has no other function.
- **Employment office** The place you want to go to apply for work.
- **Clothing store** A store that sell clothes.
- **Internet cafe** A place that offer online poker and good coffe.
- **Electronics store** A store where you can buy different things for your house.
- **Groceries store** A store that sell food that you can use to cook a homemade meal.
- **Bank** At the bank you can deposit and withdraw money. (You might get robbed if you walk around with too much money)

Continue on to Housing and Rent ---->

housing_and_rent.html:

Housing and Rent

Rent

When you start the game you live at "Small house" and have 4 weeks of rent pre-paid. Every 4th week you need to pay rent at the "Rent office". You will be notified in the weekend report when the rent is due.

Housing

Every weekend your avatar is automatically moved back to your home place and given a weekendreport. One can choose to change home place by rent another house at the "Rent office". There are some differences between the two houses. "Big house" have a better security system, so there is less chance of burglars breaking in. It's more luxurious and comfy, but it is also more expensive to rent.

Continue on to Work ---->

work.html:

Work

Applying for a job

When you start out in the game you do not have a job. In order to get a job you need to go to the "Employment office" and apply for a work position. When you click "apply for a new job" the places in the game that offer work are listed. You then click on a place you believe that you got a chance on getting a job. The work positions than is listed together with the salary the work position gives.
If you do not get the job, you will be shown a list of criterias that you did not meet. The work positions that you meet the requirements for early on in the game are typically at the places that offer low salary work positions. (For instance Fast-food store)

Working

When you have applied and gotten a new job you need to start working to earn money. You need to move to the place where you got the job, and press the work button that now appears there. Each time you press the button you work, and earn money and spend time.

Working too much

You must not become a work addict! Work addicts are often unhappy, and it could also result in injuries and medical bills.

Work experience

Each work position give a certain level of work experience. You need to work at a place for a certain amount of time before getting the work experience. In this game there are no such thing as different kinds of experience, you only get more experience by working at the "best places available".

Continue on to Food ---->

food.html:

Food

Where can I buy food?

You need food in order to survive. If you do not eat for a week, you will get a time penalty the following week. The places in the game that offer food are the Groceries store and the Fast-food store. In order to buy good at the groceries store you need to own a fridge. The benefits from cooking a meal at home versus eating junk-food are that you spend less time traveling, and it's much more cozy.

Continue on to Furniture ---->

furniture.html:

Furniture

Furniture

You buy furniture at the "Electronics store". In this release of the game there are only three kinds of furniture you can buy. The fridge is the only furniture that have any practical use besides boosting your happiness.

Continue on to Education ---->

education.html:

Education

Where do I get an education?

In order to become more educated you need to take classes at the University. Each time you press a course you wanna attend, you pay the fee and will often be given a question. If you pass the question your skill in the chosen course will increase. There are a lot of jobs in the game that require different levels of education.

Continue on to Assignments ---->

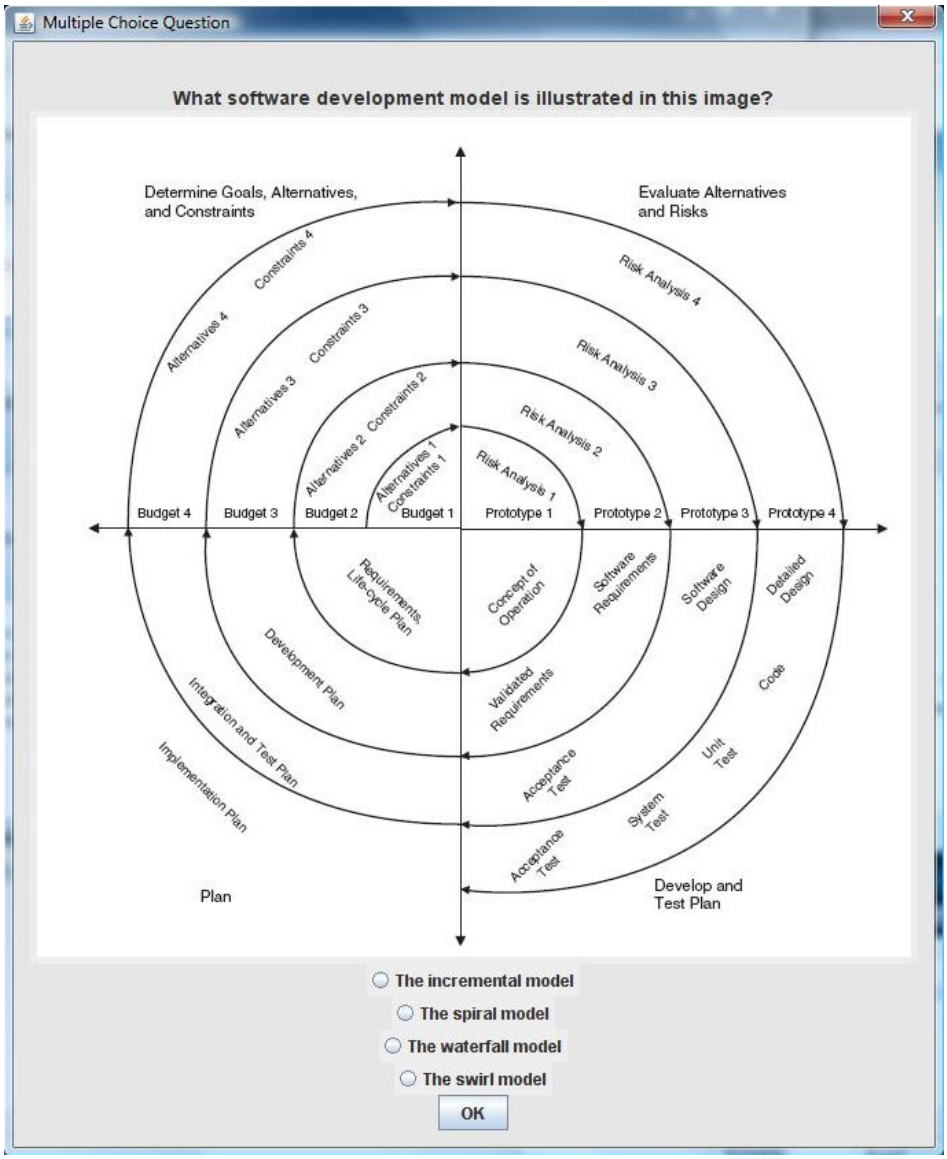
assignments.html:

Assignments

Assignments

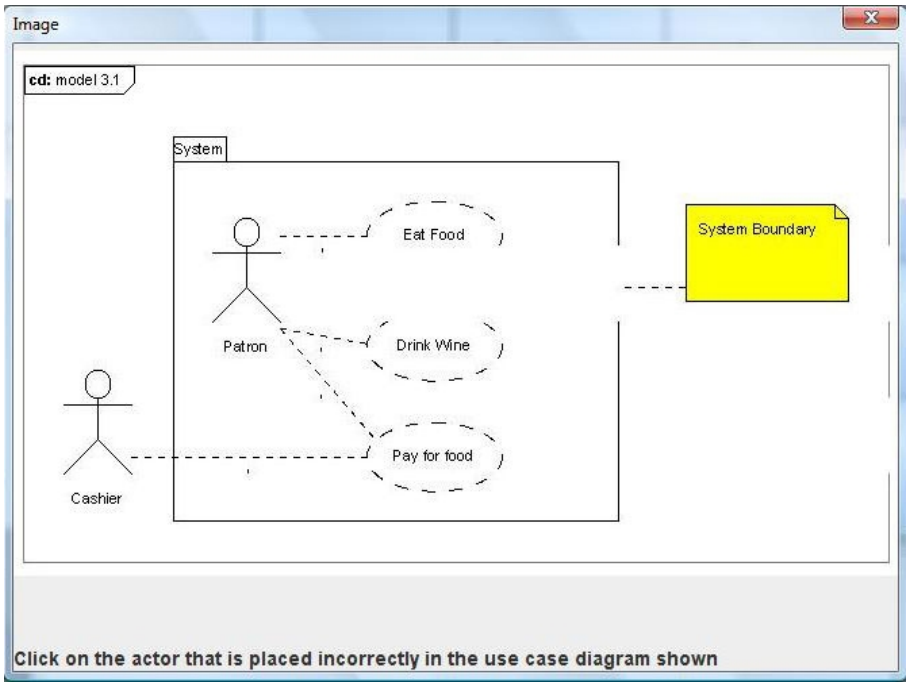
When you take a course you are often given an assignment to complete. In this game we have three different kind of assignments.

Multiple choice



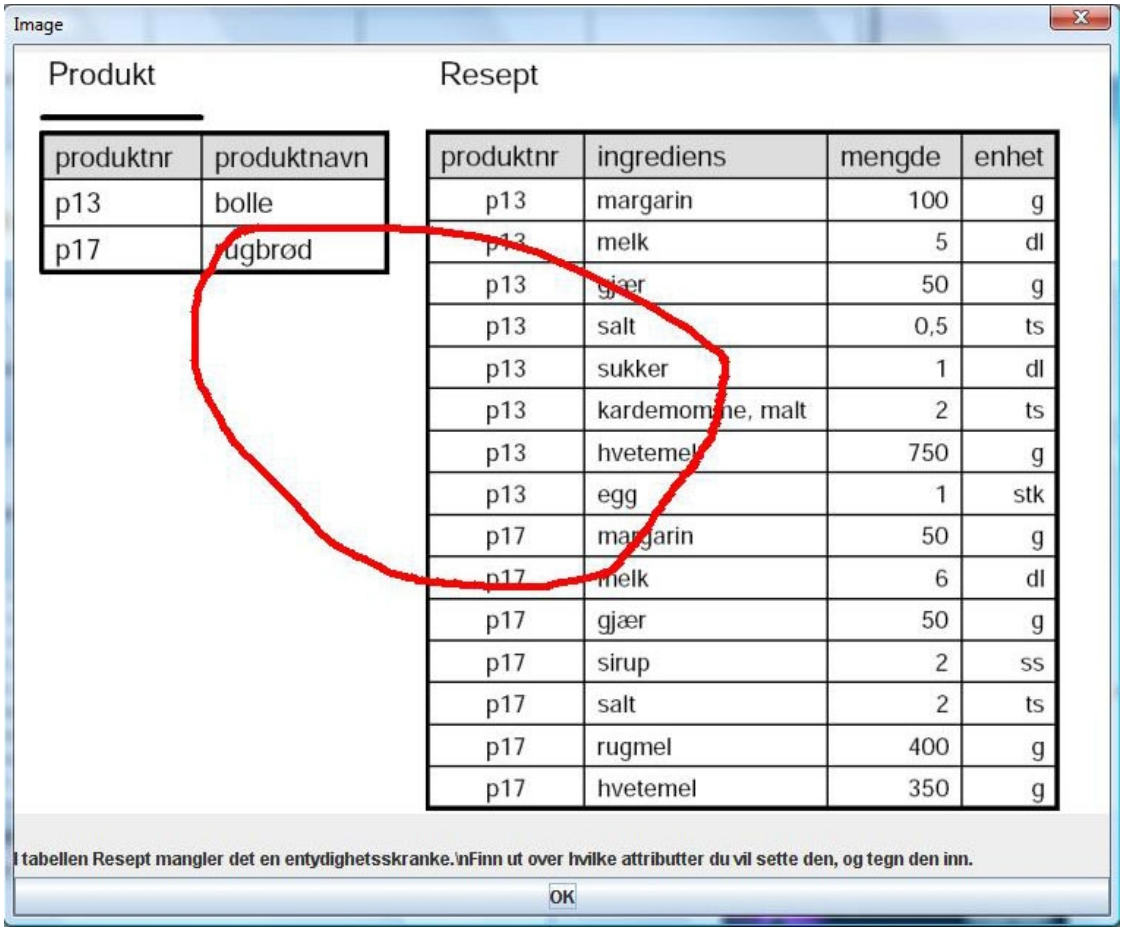
This image is quite selv explaining. You need to mark the answer you believe is the right one, and press OK.

Click on image



In this assignment you need to move your mouse pointer to the desired point on the image and press the left mousebutton to give your answer.

Draw on image



In this assignment you need to use the mouse to draw a line on the image. You start the line by moving the mouse pointer to the desired location. Then you press and hold down the left mouse button and move the mouse along the path you want to draw. Then release the left mouse button when you are at the end point of your line. If you are not satisfied with the line you drew you just left click on the image to draw a new line. When you are satisfied you click "OK" to give your answer.

Continue on to Bank ---->

bank.html:

Bank

The bank

Why do I wanna use the bank? It's obvious, if you get mugged, you won't loose all your money. In addition the bank gives you interests on the money you deposit in the bank.

Continue on to Clothing ---->

clothing.html:

Clothing

Why do I need clothes?

You will have a hard time in social settings like working etc if you do not have clothes. In this game you can only own one set of clothes, and they will become worn out and ruined if you do not buy new ones from time to time. Clothes can be bought at the clothing store. You can see what kind of clothes you are currently wearing by looking at the smiley that represent your avatar.

The different icons and what they represent

 **- Nude**
Your avatar is clothless and you will have a hard time in social settings. For instance you will not be able to work.

 **- Tweed jacket**
The clothes you start out with.

 **- Casual suit**
Better clothes than tweed jacket, but you are still not James Bond.

 **- Business suit**
If you have this suit, you will not have problems with clothing requirements at work etc.

Continue on to About the game ---->

about_the_game.html:

About the game

About the game

It is a proof of concept project and bugs might occur.
This game is a part of a master thesis written by Eirik Molnes and Jørgen Stikbakke at the

University of Oslo.
Our goal is to look more into the educational value of using educational games at university- and highschool-level.
Please contact us for further information.

About the authors

- **Eirik Molnes** - eirikmol at ifi.uio.no
- **Jørgen Stikbakke** - jorgesti at ifi.uio.no

Back to Index ---->

Appendix B
Preliminary testing sheet

Preliminary testing

This is an early alfa of our system development game. Our goal with this project is to find out the potential for learning based games in higher education. With this preliminary testing we want to see how users interact with the interface of the game, what parts of the game are understandable/unclear and what they think about the potential for the game. This test will not focus on the learning aspects of the game but rather the game-play and the user interface.

Brief overview of the game:

The game is a career based single player game. You start of with a limited amount of money, an apartment that is prepaid for one month and without a job. The gameplay evolves around climbing a career ladder through work experience and education while managing your income and expenses. You move around in a simple board game like interface and each action you perform deducts time from the ingame time. The challenge of the game is (or eventually will be) managing money, time, happiness, education and appearance of success, while being challenged by educational mini-games, to quickly climb the corporate ladder.

How these tests will be performed:

We will perform the test on two individuals with different backgrounds. One master student from IFI and one student that has just finished or about to start taking the inf1050 course. The reason we are picking two persons with different backgrounds is to get two different views on our program.

We will be present together with the test person at the time of testing. The test person will be given a laptop and a brief explanation on what the game is about and this text explaining more about the tasks of the tester. Then the tester will be asked to play the game for about 15-20minutes while expressing his/her thoughts orally.

Things you should think about while testing:

- This is not the final release of the game, there is still much work left to be done, but please comment on things you think needs improvement.
- Is there anything you feel is lacking in the user interface? Anything that would improve your game-play experience in any particular way?
- The game is meant to be a proof of concept, and does not and probably never will look particularly appealing to the eye.
- What is needed of the game to be sufficient as a proof of concept? How advanced and entertaining must the game-play be? How evolved/complex does the educational content need to be?

Your tasks as a tester:

- Play the game, while verbally expressing your opinions on the general game-play and usability of the game.
- Try to test as many aspects of the game as possible.
    - Apply for different jobs
- Work at different locations to earn money
- Test the different forms of food
- Try the different forms of questions related to education
- Buy accessories and clothing
- Rent the La Casa apartment
- See if you can find any bugs, or do anything to make the game crash.

Questions:

- What is your immediate reaction to the user interface?
- Did you understand what you where supposed to do in the game?
- How would you describe the game in terms of ease of use?
- Did you like the game as a game to play, or where you mainly focused on the questions related to system development?
- Which game elements did you enjoy, and which did you find less engaging?
- Do you think the educational mini-games are good enough for people to learn from them?
- Please explain what you liked about them and/or what you did not like about them
- Do you think that the educational aspects of the game were implemented correctly in

the game?

    - Do you believe the educational aspects of the game are balanced naturally into the game? Should there be more educational content in several parts of the game? Less?

- Did you feel that the educational questions were a natural part of the game?
- Did they feel intrusive, or entertaining?
- Please explain what you think about the four different question types (multiple choice, click on image, draw on image and SQL). Do you think any of them need any improvements?

Appendix C
The main test

index.html:

An educational system development game

Info

We are two master students at Institutt for Informatikk (IFI) at UiO that are looking more into the educational value of computer games at a university level. To test this we have made a little game with content that aim to teach the player system development, more directly, INF1050 curriculum related knowledge.

Please keep in mind that this game is developed as a 'Proof of concept. It is not optimally balanced and bugs might occur. Therefore we ask you testers to overlook small flaws if you find them, and rather think of the big picture. Could this be something you would play? Could you potentially learn anything from playing this game? Would this game make you spend more time on Inf1050 and through that learn more?

How to download, extract and play

We've developed the game in Java, so you need to have java installed in order to play the game.

Install instructions:
1. Download 'the game' to an appropriate place on your local machine.

2. Extract the zip file.
Linux extraction: unzip edugame.zip
Windows extraction: Use WinZip, WinRar or a similar program.

3. Run the file: edugame/theGame.jar
On linux: java -jar theGame.jar
On Windows: Doubleclick the file

4. Enjoy the game. We also reccomend to read through the tutorial linked at the bottom of this page.

5. Please do our survey
We will pick out one participant that will get 5x flax-lodd as a thanks for participating.

Thanks in advance!

Links

The game
The survey
Tutorial for the game

Authors

Jørgen Stikbakke (jorgesti paa ifi.uio.no)
Eirik Molnes (eirikmol paa ifi.uio.no)

Oppdatert 17. februar 2008 av Eirik Molnes

Survey - Educational system development game

Survey - Educational game inf1050

Thank you for testing our game and participating in this survey. As a reward for participating we will pick out one random person that participated and reward him/her with 5x 'flax lodd'. Therefore it is important that you write down your email adress (we prefer UiO email adresses) so we can get in contact with the winner. You can also complete the survey anonymously, just fill inn 'Anonymous' in the email field, but anonymous participants will not be a part of the lottery.
PS: These email adresses will be used to no other purpose than contacting the winner.

Email:

General questions about educational games

Question 1 What do you think about the potential for using educational games at a university level of education?

Question 2

Have you had any previous experience with educational computer games, and did you learn anything from playing them?

Question 3

How many times have you played educational computer games prior to testing our project? (One = never, Seven = a lot)

1 2 3 4 5 6 7

○ ○ ○ ◉ ○ ○ ○

Questions related to our game

Question 4

Would you play this game if it was complete with INF1050 related assignments made by the group teachers? Please elaborate on why or why not.

Question 5

How would you rate the entertainment value of our game?

1 2 3 4 5 6 7

○ ○ ○ ○ ○ ○ ○

Question 6

How would you rate the learning potential in the assignments related to Inf1050? (The ones that you found at the 'University' under the 'System development' course)

1 2 3 4 5 6 7

○○○◉○○○

Question 7

Did you learn any INF1050 related knowledge while testing our game? If you did, what did you learn?

Question 8

How would you rate the learning potential for the game on a general basis?

1 2 3 4 5 6 7

○○○◉○○○

Question 9

What is your personal highscore in the game? (The score you get in the pop-up window at the end of the game.)

Question 10

Were there anything you found particularly diffucult to understand or hard to do in the game?

Question 11

Were there any elements in the game you liked particularly good? Any things that encourged you to play more?

General feedback

Question 12

Please use this field to give your general feedback about the game. This field can also be used to report any bugs or difficulties you encountered.